大众点评网资深前端工程师周遥和
聚划算资深前端工程师李春城联袂奉献



HTML 5 网页开发实例详解

最全的HTML 5技术书，最全的HTML 5案例书








周遥 李春城 编著

- Google、苹果都支持HTML 5标准了
- BAT互联网三巨头也用HTML 5了
- 本地存储、设备访问、多媒体、平面和三维效果、性能和集成、CSS 3，这些技术核心HTML 5都有了
- 响应式设计、移动端框架、MVC、Node.js、游戏库、图形库、兼容、标准、跨浏览器，这些例子都有了


本书全部源代码

清华大学出版社

HTML 5
网页开发实例详解

周遥 李春城 编著

清华大学出版社
北 京

内 容 简 介

本书从实际的应用场景出发，结合当下的热门技术，深入浅出地介绍了HTML 5所包含的各项新技术。本书分为14章。第1~4章介绍了HTML 5和浏览器的发展史、HTML 5新特性的使用，最新的前端设计概念和第三方流行应用框架，如响应式设计、移动端框架、MVC、图形库、游戏库、Node.js；第6~12章介绍了表单、Canvas、多媒体、地理、拖放、存储、通信、离线应用等多个方向，并给出了大量实例；第13~14章通过两个完整的大型应用实例，详细分析HTML 5的项目流程及设计技巧。

本书适用于所有前端开发初学者和网页设计入门者，也可作为大中专院校及培训学校教材及上机指导用书。

图书在版编目（CIP）数据

HTML 5网页开发实例详解 / 周遥, 李春城编著. —北京：清华大学出版社，2014
ISBN 978-7-302-36136-7

I. ①H… II. ①周… ②李… III. ①超文本标记语言—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字（2014）第069756号

责任编辑：夏非彼
封面设计：王　翔
责任校对：闫秀华
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京天颖印刷有限公司
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　　张：**33　　**字　　数：**845千字

版　　次：2014年6月第1版　　**印　　次：**2014年6月第1次印刷
印　　数：1~3000
定　　价：75.00元

产品编号：057651-01

前　　言

你还在用Flash吗？“帮主”早不用了

乔布斯生前在公开信《Flash之我见》中预言：像HTML 5这样在移动时代中创立的新标准，将会在移动设备上获得胜利。

——国际巨头Google、苹果等都支持HTML 5标准，要不要学，你看着办！

BAT三巨头都偷偷用上HTML 5了

HTML 5目前在国内的发展达到了空前的高度，以BAT三大巨头互联网公司为例，它们都已经争先恐后地将HTML 5的新技术融入到了现实的开发领域中。本书的例子会涉及WebQQ、一淘网、大众点评网等公司在HTML 5方向的技术运用情况。

——还不知道BAT是谁？太out了，百度、阿里、腾讯，地球上的国人应该都知道吧。

HTML 5做了哪些改变

语义性、本地存储、设备访问、连接性、多媒体、平面和三维效果、性能和集成、CSS 3是HTML 5技术的核心，本书不光介绍八大特性的理论，重点是通过实战示例让读者精通它们。

——世界上有八大奇迹，HTML 5也有八大特性。

本书真的适合你吗

本书帮你从HTML 4时代过渡到HTML 5时代；本书提供现实生活中的应用，包括移动应用和普通PC应用；本书会涉及HTML 5的游戏库、图表库、框架介绍和案例；本书从现实的表单使用场景出发，解决低版本浏览器的兼容问题；本书介绍各种W3C规范来自什么标准，用向何处；本书提供多套作者自己实际应用的跨浏览器的原生解决方案。

——怕HTML 5不兼容？没关系，本书给出了优雅降级和各种跨浏览器兼容方案。

本书涉及的技术或框架

Modernizr
jQuery
Paper.js
WebGL
SVG
WebRTC
Socket.IO
MVC
Sencha Touch
jQuery Mobile
JQ.Mobi
HTML 5 Boilerplate
Less Framework
Raphael
Highcharts
Three.js
Sublime Text 2
Node.js
Chrome浏览器调试
Fiddler加速
JSONP
JSON
Google 地图
Swig模板
Consolidate.js

本书涉及的示例和案例

当前天气的APP
新闻阅读列表APP
一个网站的用户增长曲线图
网页中的3D效果
用Node.js搭建Web Server
销售数据图表
带字幕的视频播放器
手机定位
可拖放文本阅读器
聊天室
排队处理订单
离线留言
新消息提醒
WebQQ的通讯
花开花落的动画
MP3播放器
追踪用户位置
用浏览器拍照和摄像
类iPhone鼠标拖动效果
文件拖放上传
桌面提醒工具
微博信息实时推送
在线代码编辑器
预览网页内容
手机遥控PPT
响应式新闻阅读列表

本书特点

1. 本书不论是理论知识的介绍，还是实例的开发，都从实际应用角度出发，精心选择开发中的典型例子，讲解细致，分析透彻。

2. 深入浅出、轻松易学，以实例为主线，激发读者的阅读兴趣，让读者能够真正学习到HTML 5最实用、最前沿的技术。

3. 技术新颖、与时俱进，结合时下最热门的技术，如Node.js、响应式设计、移动开发、MVC，让读者在学习HTML 5的同时，了解熟识更多相关的世界先进技术。对于一些无法全面讲解的框架，给出了GitHub的详细地址供读者参考。

4. 贴近读者、贴近实际，大量成熟第三方组件和框架的使用和说明，帮助读者快速找到问题的最优解决方案，书中很多实例来自作者工作的大众点评网。

5. 贴心提醒，本书根据需要在各章使用了很多“注意”、“说明”等小栏目，让读者可以在学习过程中可以更轻松地理解相关知识点及概念。

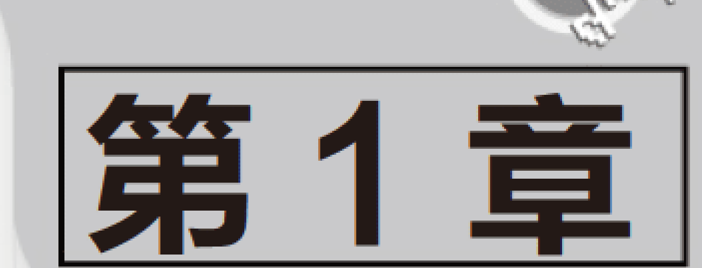
本书读者

- HTML 5开发初学者和前端爱好者
- 前端开发工程师
- 从事后端开发但对前端开发有兴趣的人员
- 想把网站移植到HTML 5技术上来的网页设计人员或站长
- 大中专院校及培训学校的学生
- 从HTML 4向HTML 5过渡的开发人员

本书配套源代码下载地址：http://pan.baidu.com/s/1o6wiaTk，若下载有问题，请电子邮件联系booksaga@163.com，邮件标题为“求代码，HTML5实例”

作 者
2014年1月

目　录



HTML 5引发的Web革命

作为一位长期从事互联网Web开发人员，在经历了早期单一的Web开发，到现今群雄逐鹿的互联网标准时代后，内心肯定迫切需要一项统一的Web标准。HTML 5正是具备担负这一伟大历史使命的最佳候选，也正是它才能结束这一时代，开辟HTML的新纪元。

从1993年HTML诞生以来，HTML（超文本标记语言）逐渐成长为当今网络应用最广泛的语言，它的易用性、扩张性和与平台无关性成为其在网络技术应用中的杀手锏。但随着移动互联网的发展，原有的技术特性被消耗殆尽，开发成本变得越来越高，技术实现难度越来越大。传统的Web开发似乎正在慢慢从宝座上褪去光环，一点点被蚕食，逐渐沦为一个配角。时间倒回到2004年，一小组人成立了WHATWG（Web Hypertext Application Technology Working Group，Web超文本应用技术工作组），开始致力于开发HTML 5。2008年HTML 5的第一份草案由伊恩·希克森撰写。直到今日，经历了多年的发展和沉淀，已经初步形成以语义性、本地存储、设备访问、连接性、多媒体、平面和三维效果、性能和集成、CSS 3八大技术特征为一体的新一代HTML。“一次编写，到处部署”这一技术梦想，似乎离我们已经越来越近，Web开发人员的又一个春天即将到来。

在HTML 5的发展过程中，不得不感谢两家公司。第一家是乔布斯的苹果公司，2010年乔布斯公开声明封杀Flash，声称“Flash是为了使用鼠标的PC而设计的，不适用于用手指操作的触屏设备”，在苹果的触屏设备上永远不会出现Flash，这使得当下很多公司把目光移向HTML 5。另外一家是谷歌，它开发和设计了简洁、高效的Web浏览器Google Chrome，并发布了一系列以Gmail、Google Docs为代表的互联网应用，让人们认识到原来Web应用也可以如此的高效和丰富。

HTML 5已经经历了10个年头的发展，但它不是一项新的技术，是对HTML标准的改进和加强，现有的浏览器应用已经越来越多地加入了HTML 5特性，我们有理由相信在下一个10年，它仍旧会带着HTML的基因成为网络霸主，让我们拭目以待。

本章知识点：

- 了解HTML 5
- 认识主流浏览器
- 制作简单的HTML 5页面
- 检测浏览器对HTML 5的支持情况



1.1 你是不是真的了解HTML 5

如果读者认为HTML 5是一项与传统HTML不同，甚至是需要从零开始学习的一项技术，那就大错特错了。HTML 5并没有完全地颠覆HTML，而是对HTML一部分繁冗的特性进行简化，对不足的特性进行补足和加强。不用丢掉以前HTML的标签，也不需要重新学习以往的知识，就可以使用HTML 5。举个简单的例子，将原有的HTML应用升级至HTML 5，只需要在每个文档的第一行声明“<!DOCTYPE html>”，只有这一种格式，不需要定义不同的DOCTYPE，这正体现了HTML 5化繁为简的特色。

HTML 5的新特性可以分为八大类，如图1.1所示。



图1.1 HTML 5的八大特性

虽然HTML 5仍处于完善当中，但是绝大部分浏览器已经实现了其中绝大部分特性，千万不要错过这一技术变革，一起行动起来，成为合格的HTML 5布道者。

1.1.1 通过W3C认识HTML 5的发展史

HTML首次于1993年以因特网草案的形式发布，经历了2.0版本、3.2版本、4.0版本直到1999年的4.01版本的演化和发展，慢慢由W3C掌握了HTML的标准和规范制定。W3C全称为World Wide Web Consortium，指的是万维网联盟，于1994年10月创建，致力于实现Web标准化，其中最著名的标准规范包括XML、CSS、HTML等。

HTML经过多年的发展，直到Web 2.0的提出，被重新提到一个新的高度。传统的静态网站慢慢被一些动态网站所替代，其中最著名的技术莫过于Ajax。越来越多的开发者认识到，通过浏览器也能够实现与PC软件相媲美的应用，此时出现了一大批Web 2.0的网站，如现今的绝大多数网站，如图1.2所示。



图1.2 Web 2.0网站

Web 2.0虽然美好，但也有其技术的局限性，如多媒体的应用、数据的传输等方面仍然显得势单力薄，广大的Web开发者期盼出现一个更强的标准，此时HTML 5的出现更像是一剂强心剂，开发者不需要丢弃以往的开发经验，也不需要重头再开始学习，就可以实现以往无法实现的功能。据维基百科记录，HTML 5的前身是Web Applications 1.0，该草案在2004年由WHATWG提出，并于2007年获W3C接纳。之后新成立的HTML 5团队在2008年的1月22日推出第一份正式草案。从2009年开始，各大浏览器陆续开始支持HTML 5的部分特性，直到2012年12月17日，W3C宣布HTML 5规范正式定稿，至此HTML 5修成正果。可以通过图1.3了解整个Web技术变革的时间线。

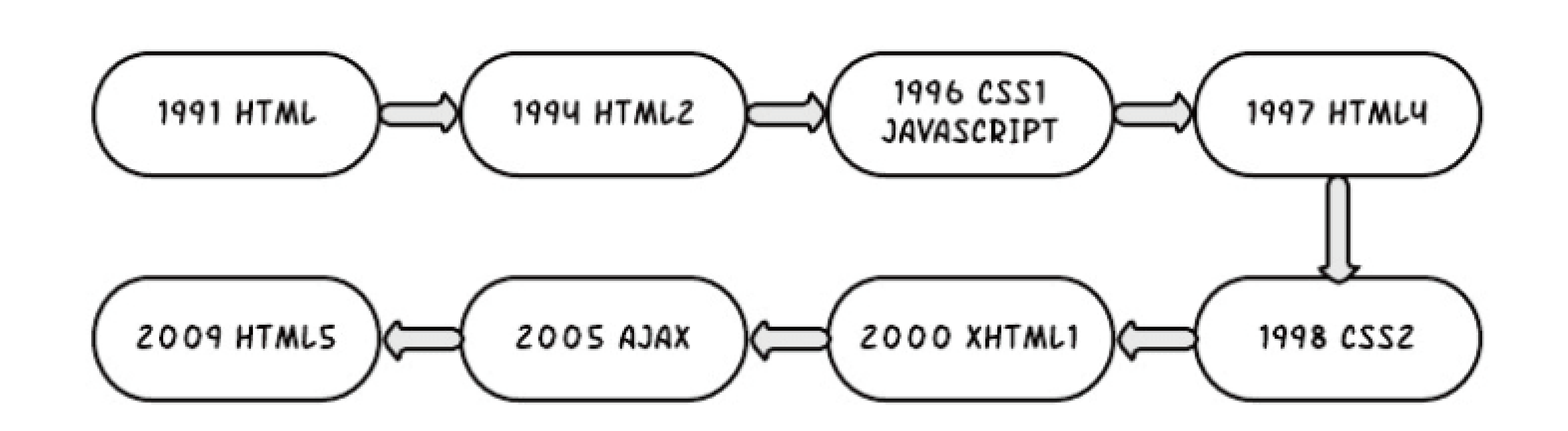


图1.3　Web技术变革的时间线

虽然HTML 5近两年非常火爆，常常出现在人们的视线中，但任何一项技术从出现到发展，直到最后被广泛接受都不是一件容易的事情，期间需要度过种种困难，有些甚至已经超越了技术自身的范畴，尤其是在幕后的一些国际互联网巨头，起到了至关重要的作用。

近年来，W3C也开始出现一批国内互联网公司的身影，如2011年1月11日，腾讯公司加入W3C，将共同参与包括Web APP、HTML 5等新互联网技术标准的研究和制定；2011年9月25日，百度也加入W3C。越来越多的中国互联网公司正在加入W3C，势必对推动HTML 5的发展，尤其对中国互联网的快速发展起着积极的作用。

1.1.2　HTML 4、XHTML、HTML 5的区别

从HTML历史的发展角度来说，HTML 4、XHTML、HTML 5依次出现，每个都肩负着自身的使命。

HTML 4出现之前，那是一个书写样式非常痛苦的年代，仅有的部分样式以属性的形式出现在元素节点上，文档的表现和结构被完全地耦合在一起，1997年出现的HTML 4结束了这一切，一个可以书写独立样式表，结构和表现相分离的HTML时代从此诞生。不仅如此，HTML 4还扩展了HTML的脚本、框架、嵌入对象、表格、表单，并且为残疾人提供了访问的可能。

XHTML全称The Extensible HyperText Markup Language，中文名为可扩展超文本置标语言。其1.0版本在2000年1月26日成为W3C推荐标准。XHTML是一种在HTML 4基础上进行改进和优化的新语言，它的语法更加简洁、更加严谨。与HTML4相比XHTML有如下区别（只列举部分）：

- 要求所有的标签都要闭合；
- 要求所有的元素和属性名都必须用小写字母；
- 要求标记必须有合理的嵌套；
- 要求所有属性值必须用单引号或双引号包括。

XHTML更多的是从规范角度出发，对HTML 4进行改进和优化，一段完整的XHTML代码如下：

```
<!DOCTYPE html>
<html>
    <body>
          <h1>My First Heading</h1>
          <p>My first paragraph.</p>
    </body>
</html>
```

本书所介绍的HTML 5出现的最晚，带着语义性、本地存储、设备访问、连接性、多媒体、平面和三维效果、性能和集成、CSS3八大技术特性，怀揣着“Write Once，Run Everywhere”的梦想而来，又有苹果和谷歌等国际互联网公司的支持。也许未来的传统PC桌面应用会逐步被以浏览器为核心的网络和“云”所替代，会有更多如Gmail和Google Docs这样优秀的Web App诞生。当然现在这么说还为时过早，但不可否认的是，一场技术变革正在开始，也许HTML 5不是最好的，但HTML 5一定是最适合这个时代的，是Web开发的必然选择。新一代的HTML 5语义标签如图1.4所示，其中的标签含义如下所示。

- header：标记头部区域内容；
- footer：标记脚步区域内容；
- section：页面中的一块区域；
- article：文章内容；
- aside：相关内容或引文；
- nav：导航类辅助内容。

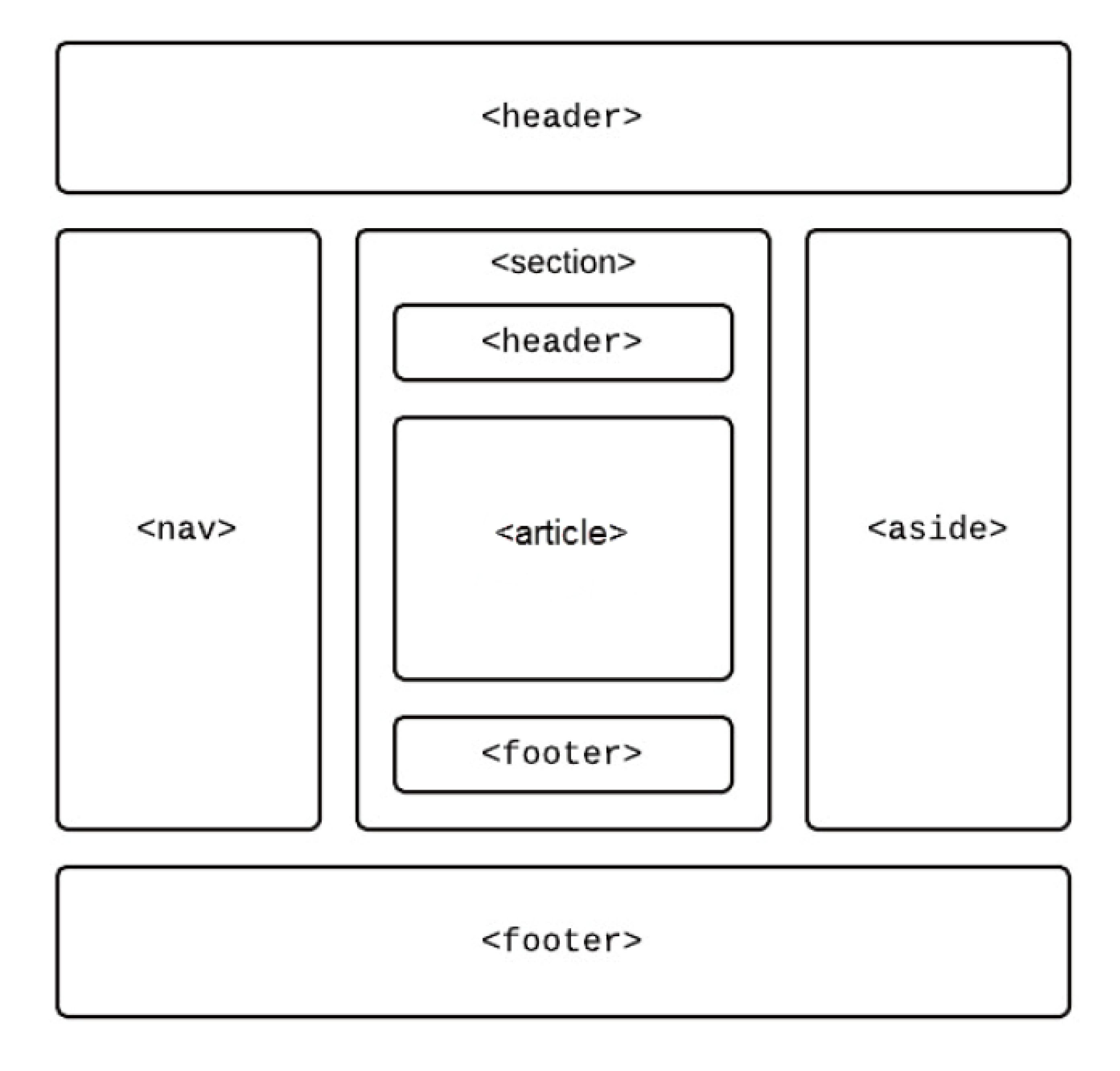


图1.4 新一代的HTML 5语义标签

1.1.3 什么人应该学HTML 5

2012年Facebook CEO扎克伯格跳出来宣布“豪赌HTML 5是Facebook最大战略错误”，似乎预示着HTML 5是一条极度漫长且不平坦的路，不过俗话说“美景总在险峻中”，越是险峻的地方越是能领略到平地上无法带来的感受。

HTML 5虽然在移动设备或一些特殊场景上使用还不是很成熟，开发成本过高，但这并不影响它带来重大改变的事实，面对这样一次变革，开发人员所要做的就是怀揣着好奇之心尽情释放自己的热情去拥抱它，HTML 5代表着未来的趋势，而且已经非常接近我们日常的开发。

图1.5是第三方收集的开发者学习HTML 5的调查数据。

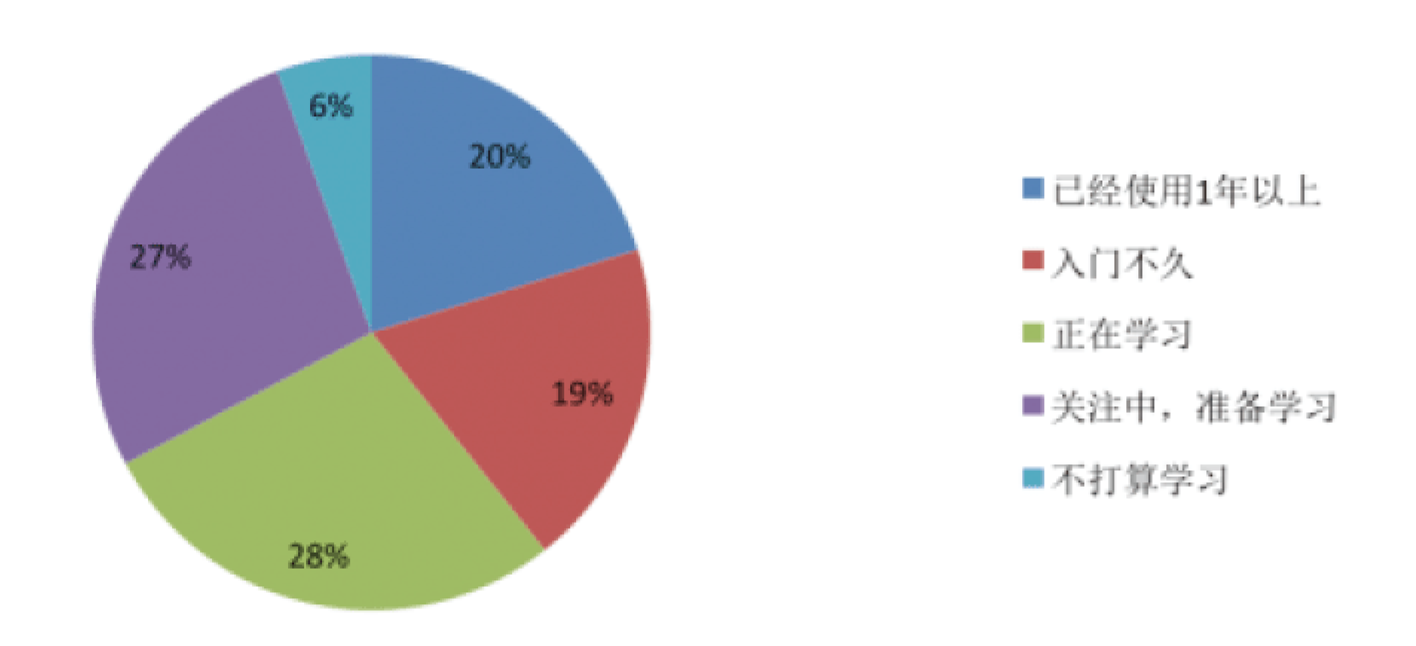


图1.5 HTML 5学习调查

如果是Web开发的新人，那么从最新的HTML 5版本开始学，比起从HTML 4.01、XHTML 1.0和DOM 2 HTML一路学习过来能更加容易抓住重点。然后可以再回过头重新看之前的HTML版本，体会HTML 5设计者的良苦用心。如果是已经从事Web开发的人员，那么学习HTML 5无疑会提升自身的技术储备和实力，成为升职加薪的砝码。如果是Web开发的老手，那么HTML 5能重新燃起对Web开发的激情。

图1.6列举了到目前为止业内认为HTML 5的优缺点，以便读者在学习HTML 5的激动之余保持冷静的态度。



图1.6 HTML 5优缺点

1.1.4 一张图告诉你如何学习HTML 5

HTML 5的学习必须从八大特性开始入手，图1.7给出了八大特性和其他相关的知识要点，可以看出HTML 5覆盖面之广，知识之庞杂。

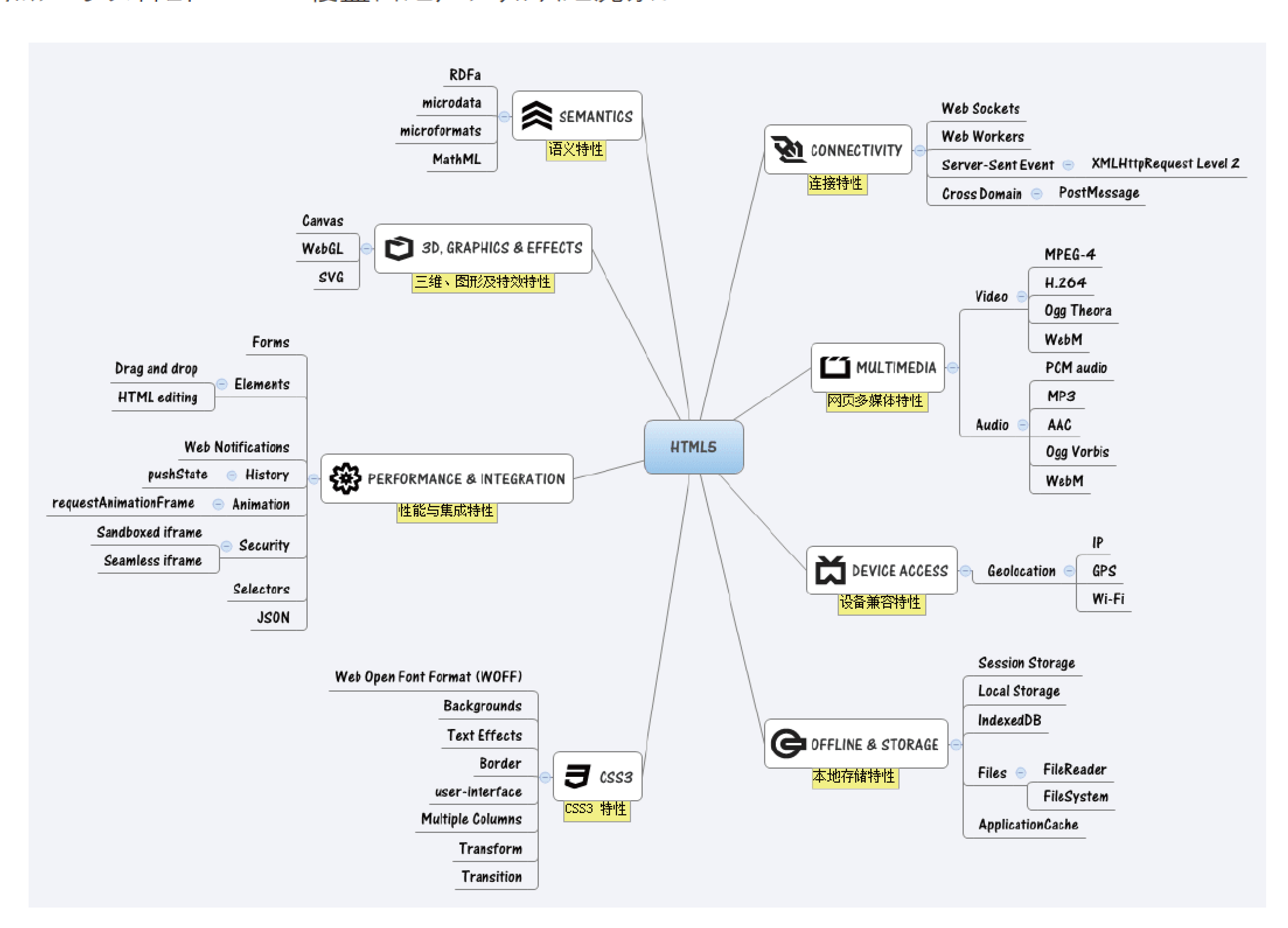


图1.7 HTML 5的八大特性和相关知识要点

读者可以参考图1.7，通过本书的说明和示例一步步学习HTML 5，体会HTML 5带来的不一样的开发体验。



1.2 浏览器之争

最早的浏览器Mosaic由美国伊利诺州的伊利诺大学的NCSA组织在1993年3月份推出，当时还只能显示一些简单的图片和图形。1994年网景推出Netscape Navigator给人们多了一种选择。微软的IE（以下简称IE）推出后，战场烽烟四起，聪明的微软早期让IE通过Windows操作系统捆绑战略一举击败了网景公司的Netscape Navigator，从而奠定了霸主地位。面对微软的强大危险，1998年网景公司无奈将Netscape Navigator程序开源，同时成立了非盈利性的“Mozilla基金

会（Mozilla Foundation）”，2002年推出了Firefox。

伴随着微软操作系统的快速普及，IE屡创新高，2004年其市场份额达到历史巅峰93%。在此期间的2003年，苹果公司发布Safari浏览器。好景不长，从2004年后IE的市场份额一步步被其他浏览器蚕食，直到2008年，谷歌公司推出Google Chrome浏览器，一路高歌猛进，2012年5月，Chrome浏览器在全球范围超过IE，一举拿下霸主地位。表1.1罗列了历年主流浏览器市场份额占比。

表1.1　历年主流浏览器市场份额

浏览器	1994年	1996年	2000年	2004年	2006年	2008年	2009年	2010年	2011年	2012年
Chrome	未出现	未出现	未出现	未出现	未出现	0.91%	3.27%	10.25%	21.08%	32.64%
Safari	未出现	未出现	未出现	0.72%	3.42%	2.7%	3.10%	4.31%	5.33%	7.23%
Opera	未出现	未出现	0.13%	0.59%	0.57%	2.7%	2.62%	1.96%	1.84%	1.71%
Mozilla Firefox	未出现	未出现	0.07%	3.44%	11.46%	25.49%	30.48%	32.27%	28.20%	23.86%
IE	未出现	7.97%	81.53%	93.25%	83.15%	67.93%	59.7%	51.45%	42.93%	33.32%
Netscape	18%	84.91%	15.94%	0.27%	0.95%	停更	停更	停更	停更	停更
Mosaic	68%	停更	停更	停更	停更	停更	停更	停更	停更	停更

1.2.1　说说这些常见的浏览器

1993年3月，Mosaic的第一个面向普通用户的预览版本发布，同年9月份支持Windows等操作系统，从此世界上第一个图形接口浏览器诞生，如图1.8是Mosaic浏览器的截图。



图1.8　Mosaic浏览器

1994年12月，Netscape浏览器的1.0版本发布，但当时互联网尚未完全普及，浏览器还属于一项比较超前的科技，没有走进大众的生活中。Netscape浏览器并没有走多远，当Window 98出现以后，微软公司将IE与操作系统进行捆绑并免费提供，从此Netscape市场份额开始下降，终于在1998年的11月将Netscape软件开源，如图1.9为Netscape最后一个版本Netscape 9的截图。

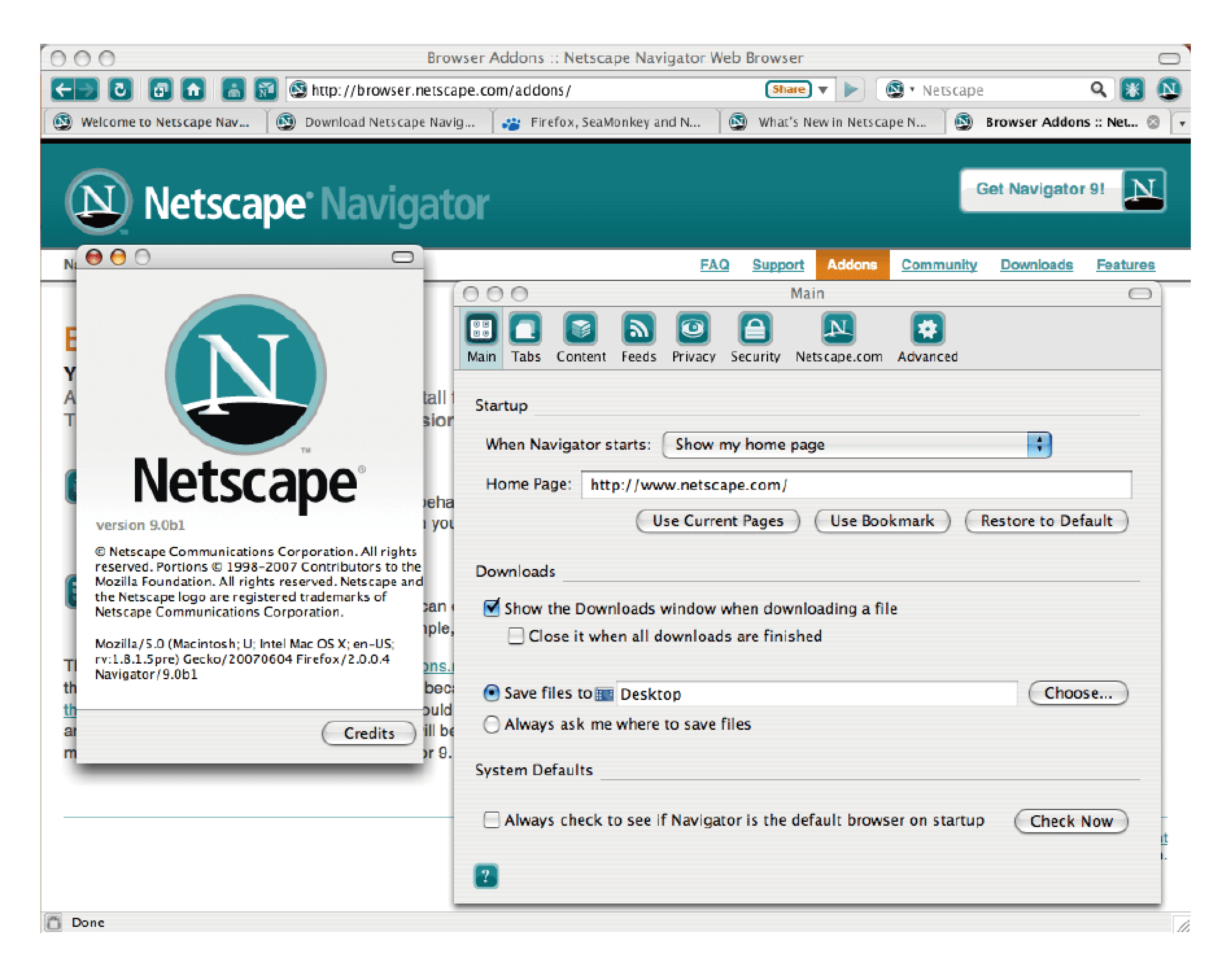


图1.9 Netscape 9浏览器

IE的第一个版本于1995年8月16日发布，从此一个大家最耳熟能详的浏览器诞生了，伴随着Windows操作系统，IE不断成长，目前最新的版本是IE 10，效果如图1.10所示。



图1.10 IE 10浏览器

Mozilla这个单词由Mosaic Killer演变而来，当时的网景公司取这个名字是想让其取代当时世界第一的浏览器Mosaic。2002年Mozilla 1.0正式发布，之后升级缓慢，但2011年由于感受到Google Chrome强大的压力，加快了升级步伐，从2011年的5.0版本经过快速迭代升级至2013年最新的Finefox21.0版本，不过市场份额却与升级速度相反，但它确实是一款优秀的浏览器，如图1.11为Firefox 21.0版本的截图。



图1.11　Firefox 21.0

Opera中文名“欧朋”，从1997年正式发布Windows的2.1版本后，一直都是一款相对低调的浏览器，不过近几年由于移动互联网的兴起和“欧朋”较早的进入，“欧朋”这个名字慢慢被大众所熟知，其优秀的UI设计和高效的性能一直是它多年来不变的特点，如图1.12是最新版本Opera 12的截图。



图1.12　最新版本Opera 12

Safari是大名鼎鼎的苹果公司开发的浏览器，于2003年发布首个测试版本，Windows操作系统的首个测试版本于2007年正式发布，不过很可惜，苹果公司于2012年的7月27号停止了对Windows的更新。相较于之前出现的浏览器，Safari浏览器设计简洁、大方，秉承了苹果一贯的设计理念，图1.13为Safari 5.1.7的截图。



图1.13　Safari5.1.7截图

最后一个出现的是Google Chrome，该款浏览器几乎集结了以上所有浏览器的特点，从2008年出现后一路高歌猛进，取得了非常傲人的成绩，同时也成为像笔者一样的Web前端人员平日开发调试的首选，同时还是本书示例展现的首选浏览器，如图1.14是Chrome 29.0.1530.2的截图。

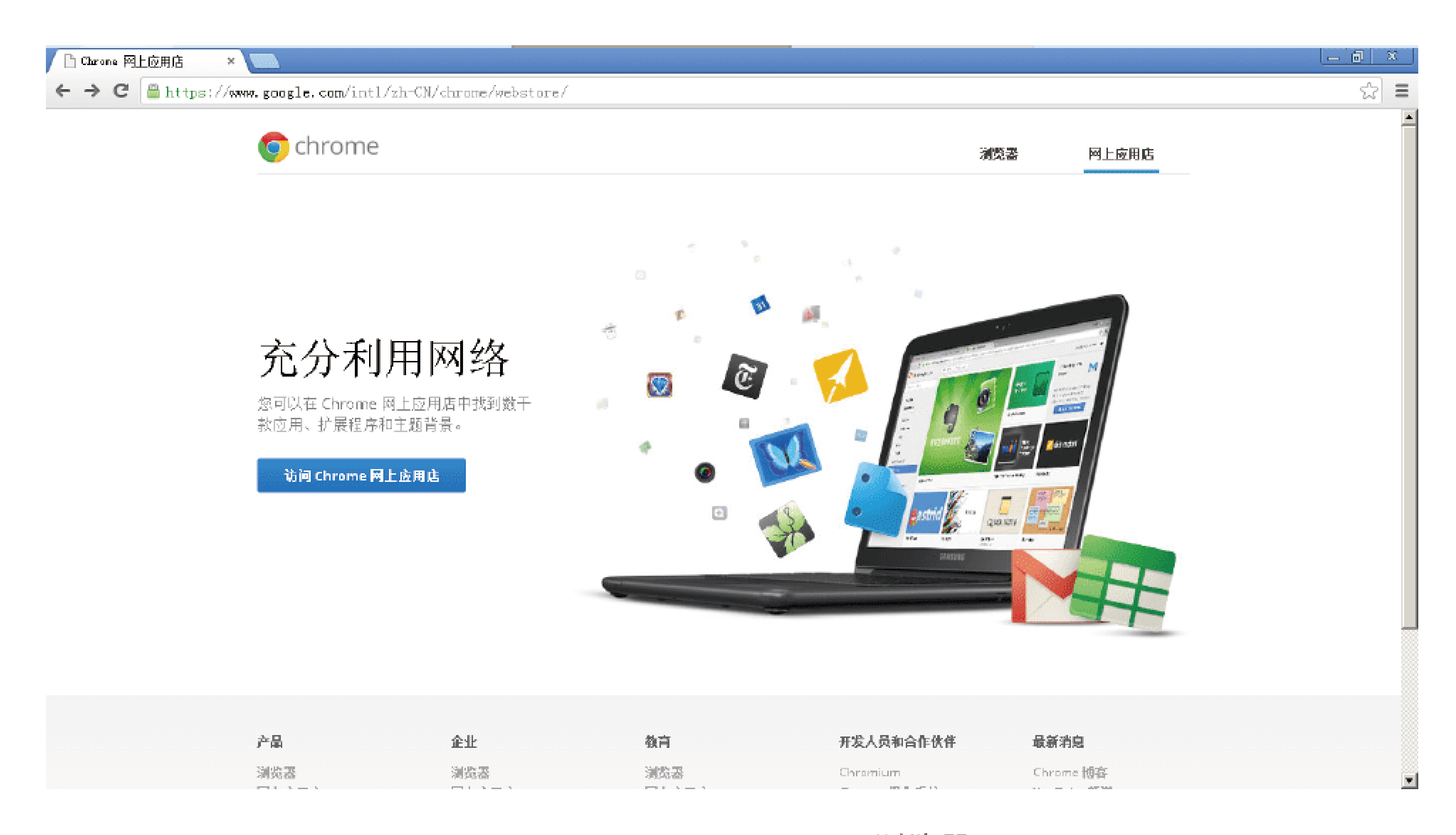


图1.14　Chrome 29.0.1530.2浏览器

看完了目前世界上主流浏览器的介绍，接着，对比各组浏览器的性能数据（数据来源于网站http://sixrevisions.com），图1.15为JavaScript在各浏览器运行的速度对比，可以看到IE表现非常糟糕。

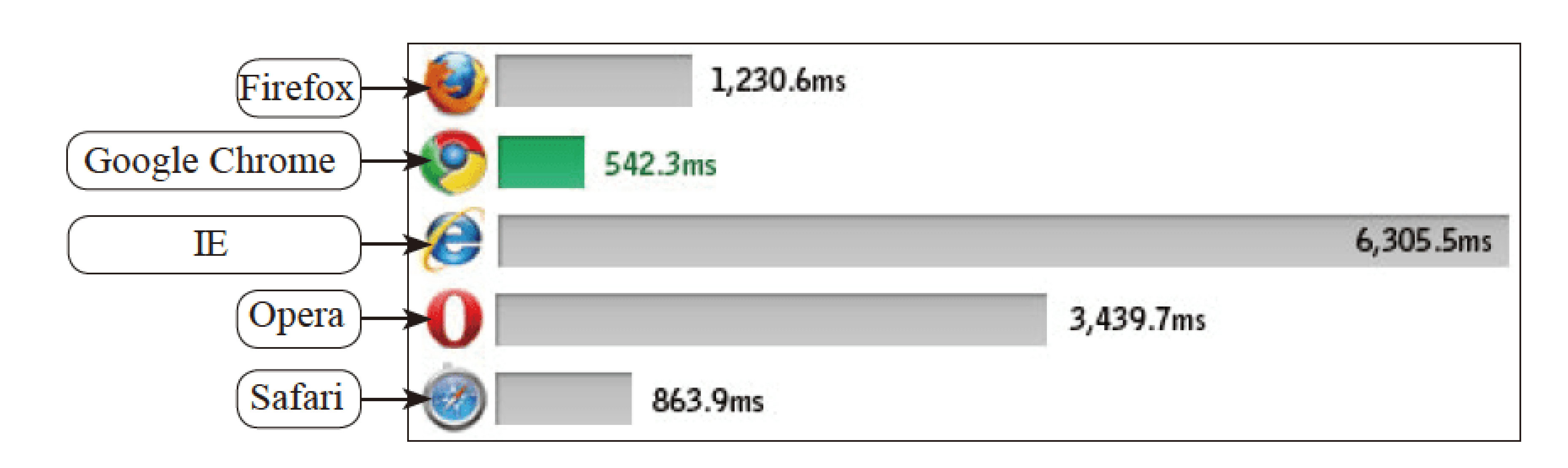


图1.15 JavaScript在各浏览器的运行速度

图1.16为浏览器CSS渲染速度对比，Google Chrome的表现非常突出，已经进入100ms以内。

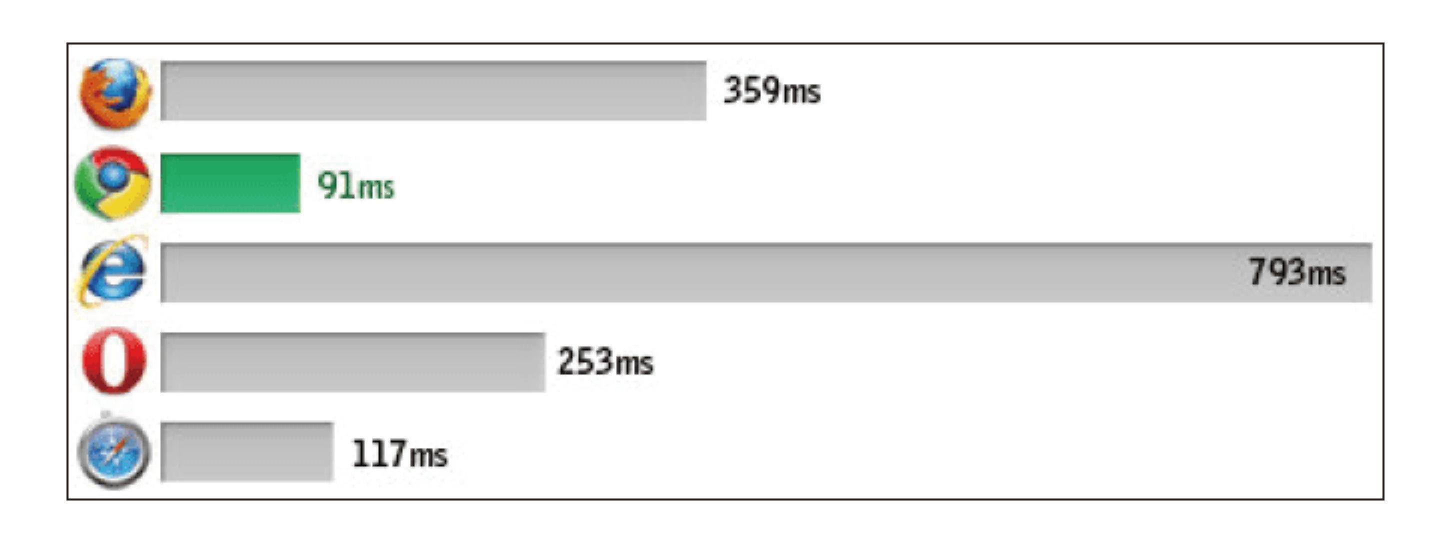


图1.16 浏览器CSS渲染速度对比

如图1.17为浏览器运行时CPU使用情况对比，还是Google Chrome表现最为出色。

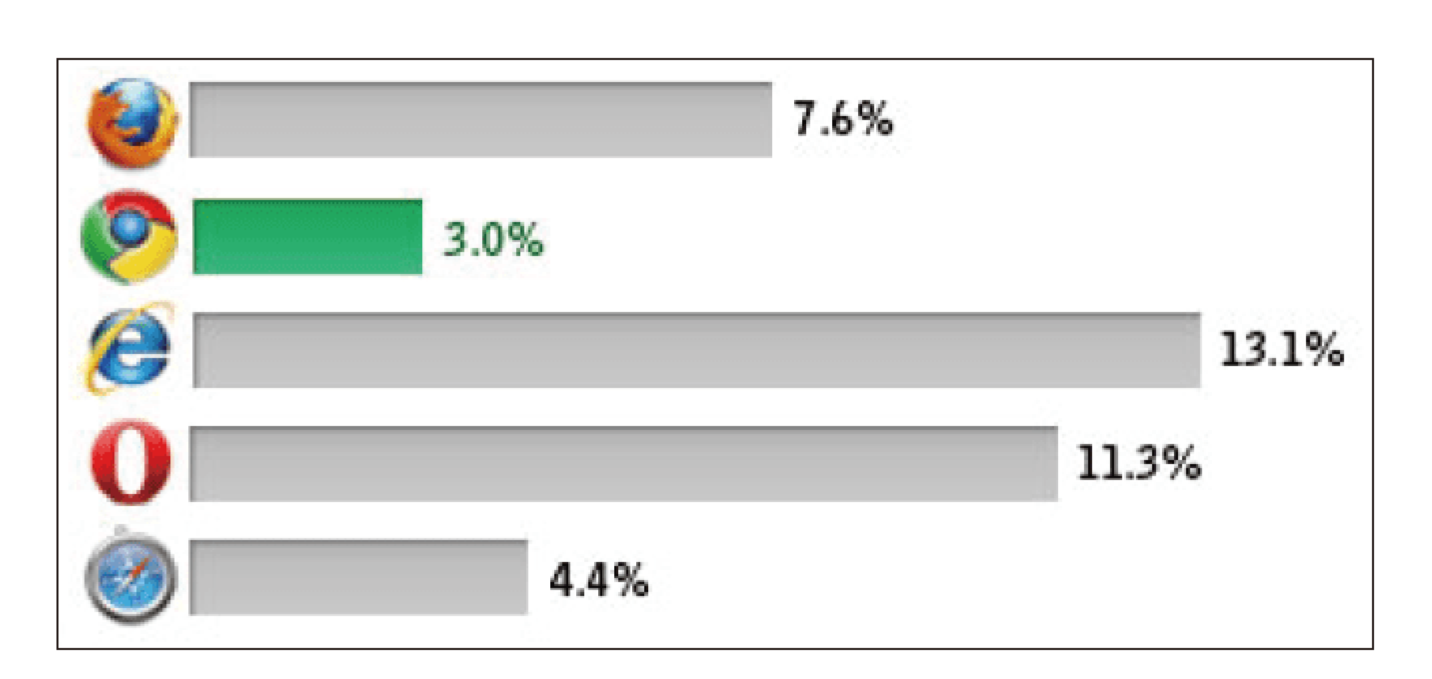


图1.17 浏览器CPU使用情况对比

综合上述数据，Google Chrome表现最佳，IE最糟糕，当然这只是目前为止的数据，期待未来各种浏览器都有更出色的表现。

1.2.2 浏览器的兼容烦恼与策略

Web前端开发是一项具有挑战性的工作，除了要学会及时自我更新世界流行前端技术知识外，还要学会应对不同浏览器的挑战。截至目前为止，一般网站需要兼容的浏览器有IE 的6、7、

8、9、10版本、Opera、Mozilla Firefox、Google Chrome、Safari，还有国内基于Trident或Webkit开发的一批浏览器，如遨游浏览器（Maxthon）、360浏览器、世界之窗（The world）、搜狗浏览器、TT浏览器等，可见，Web前端开发并非表面看起来那么轻松，而是一个相当繁杂的事情。



Trident是微软的视窗操作系统Windows搭载的网页浏览器——IE的排版引擎的名称。WebKit是一个开源的浏览器引擎，苹果的Safari、谷歌的Chrome浏览器都是基于这个框架开发的。表1.2列出了各浏览器的内核。

表1.2 主流浏览器核心

IE、遨游、腾讯TT、世界之窗、360浏览器	基于Trident内核
Safari、Google Chrome、Camino	基于WebKit内核
Mozilla Firefox、Camino	基于Gecko内核
Opera	基于Presto内核

开发者常见的需要应对的浏览器兼容性分两种：JavaScript和CSS。

（1）JavaScript兼容问题，比如想要获取元素中包含的所有文本内容，不同浏览器获取方式如下：

```
// IE浏览器获取方式
document.getElementById('element').innerText = "element text";
// 非IE浏览器获取方式
document.getElementById('element').textContent = "element text ";
```

（2）CSS浏览器的兼容问题，比如设置元素透明度，不同浏览器的实现方式如下：

```
// IE浏览器使用滤镜实现
filter:progid:DXImageTransform.Microsoft.Alpha(style=0,opacity=50)
// 非IE浏览器
opacity:0.5
```

现在的前端开发，已经出现了非常多的框架和类库用于浏览器的兼容问题，比如最常见的jQuery类库，解决获取元素中包含的所有文本内容兼容问题时，可以使用以下语法：

```
$('#element').text('element text');
```

平常在书写样式表时，若要使用CSS 3中的各种特效，都需要加上各种前缀让不同的浏览器得以支持，如实现一个圆角效果代码如下：

```
border-radius: 3px;                    /* 圆角，水平半径为3px */
-moz-border-radius: 3px;               /* Fierfox浏览器 */
-webkit-border-radius: 3px;            /* WebKit核心浏览器 */
-o-border-radius: 3px;                 /* Opera浏览器 */
```

笔者推荐一个非常流行的解决方案，使用Less CSS Framework解决书写样式的兼容性问题，以下描述摘自官网：

“LESS将CSS赋予了动态语言的特性，如变量、继承、运算、函数。LESS既可以在客户端上运行（支持IE 6+，Webkit，Firefox），也可以借助Node.js或者Rhino在服务端运行。”



LESS中国官网地址为http://www.lesscss.net/，读者可以参考网站学习，本书后面也有专门的章节介绍该框架。

1.2.3 给你的浏览器打分

给读者推荐一个HTML 5评测网站，该网站用于测试浏览器支持HTML 5新特性的情况，评测的总分为500分。按照目前网站的记录，桌面浏览器排行中排在第一的是国内遨游浏览器4.0版本，分数为476分，桌面浏览器排行如图1.18所示。

		Score
Maxthon 4.0 »		476
Chrome 27 »		463
Opera 12.10 »		419
Firefox 21 »		399
Safari 6.0 »		378
Internet Explorer 10 »	*Microsoft Surface and others*	320

图1.18　桌面浏览器HTML 5特性支持

另外，网站还提供了手机浏览器排行榜，如图1.19所示。

			Score
BlackBerry 10 »		*BlackBerry Q10 or Z10*	485
Opera Mobile 14 »		*Android*	448
Chrome 25 »		*All Android 4 devices*	417
Opera Mobile 12.10 »		*Multiple platforms*	406
Firefox Mobile 19 »		*Multiple platforms*	399
iOS 6.0 »		*Apple iPhone, iPad and iPod Touch*	386
Windows Phone 8 »		*Nokia Lumia 822, HTC 8X and ot...*	320
Android 4.0 »		*Samsung Galaxy Nexus*	297
Bada 2.0 »		*Samsung Wave and others*	283
Nokia Belle FP 2 »	S60 5.5	*Nokia 808 PureView and others*	272
Android 2.3 »		*Google Nexus S and others*	200

图1.19　手机浏览器HTML 5特性支持

通常，用平板电脑浏览网页的频率高于手机，平板电脑上的浏览器排行情况如图1.20所示。

		Score
Opera Mobile 14 »	*Android*	448
Chrome 25 »	*All Android 4 devices*	417
RIM Tablet OS 2.1 »	*BlackBerry PlayBook*	411
Opera Mobile 12.10 »	*Multiple platforms*	406
Firefox Mobile 19 »	*Multiple platforms*	399
iOS 6.0 »	*Apple iPhone, iPad and iPod Touch*	386
Silk 2.2 »	*Amazon Kindle Fire*	358
Internet Explorer 10 »	*Microsoft Surface and others*	320
Android 4.0 »	*Asus Transformer Prime and oth...*	297
webOS 3.0 »	*HP TouchPad*	224

图1.20 平板电脑浏览器HTML 5特性支持

同时，笔者还测试了本地Google Chrome的HTML 5支持情况，版本为29.0.1541.0，测试结果分数为473分，评测如图1.21所示。

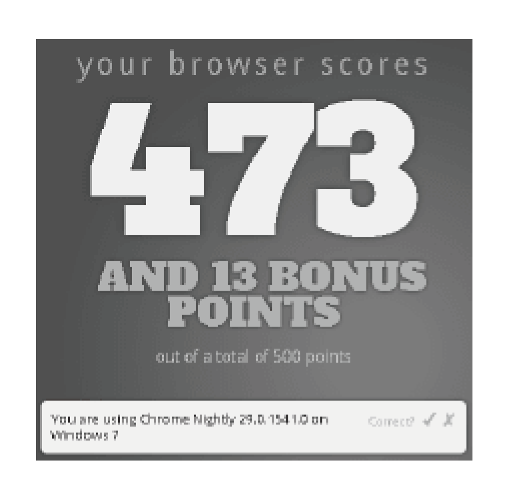


图1.21 Google Chrome 29.0.1541.0评分

国内很多新兴的浏览器厂家经常用支持HTML 5的评分制造噱头，但是，一款优秀的浏览器不只能一味求新。在给浏览器评分时，还需要针对于多个方面进行比较，如脚本执行、页面渲染、安全性、易用性，这些基础功能，尤其在当今，用户经常采用浏览器进行电子购物消费，安全性显得尤其重要。希望国内浏览器厂商能在这方面下更多的功夫，为建设安全的网络平台打下坚固的基石。



1.3 学习制作简单的HTML 5页面

HTML 5有种类丰富的语义结构新标记，除了与块元素有很多相似性外，还拥有自身的语义行为。在制作一个HTML 5页面时，首先要告知浏览器使用何种HTML或XHTML规范，在HTML 5出现之前，经常看到冗长的规范语法，代码如下：

```
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

HTML 5简化了这一约定，而且不区分大小写，DOCTYPE如下：

```
<!DOCTYPE html>
```

下面先通过一张截图观察简单的HTML 5页面，如图1.22所示。



图1.22　简单的HTML 5 页面

通过一张页面结构示意图查看页面基础构造，如图1.23所示。

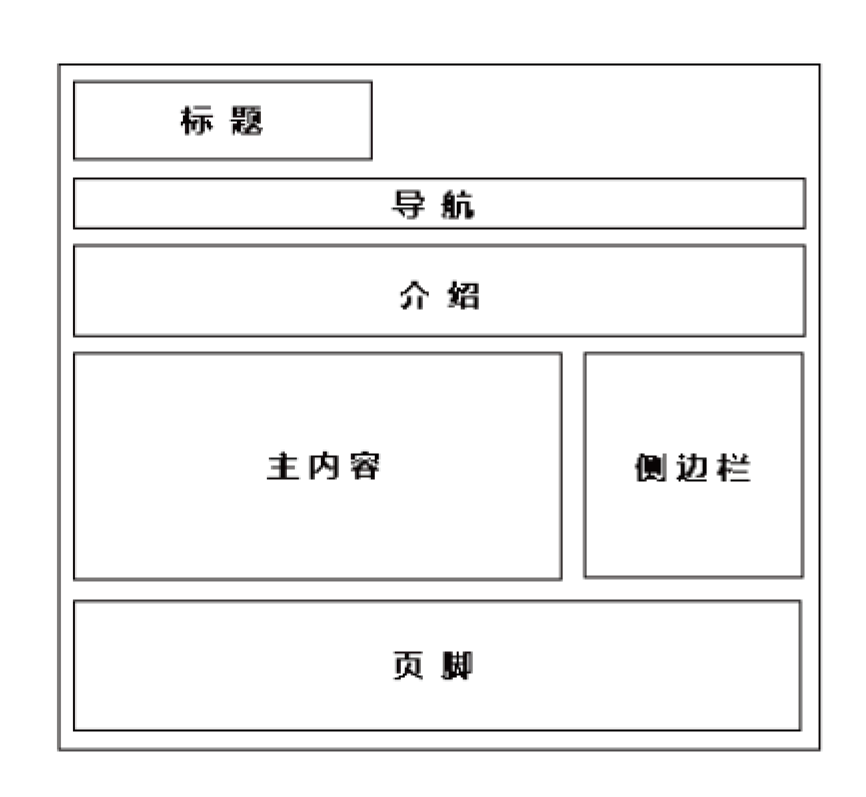


图1.23　页面基础构造

页面被分为6个区域，分别为：标题、导航、介绍、主内容、侧边栏、页脚。

（1）页面基础结构代码如下：

```
<!doctype html>
<html>
<head>
```

```
        <title>简单的HTML 5页面</title>
        <link rel="stylesheet" href="../css/blog.css" type="text/css" />
                <!-- 页面依赖样式表 -->
</head>
<body>
        <header>                                          <!-- 标题 -->
                <h1>HTML 5</h1>
        </header>
        <nav></nav>                                       <!-- 导航 -->
        <section class="intro"></section>                 <!-- 介绍 -->
        <section class="mainContent"></section>           <!-- 主内容 -->
        <aside></aside>                                   <!-- 侧边栏 -->
        <footer></footer>                                 <!-- 页脚 -->
</body>
</html>
```

（2）导航区域可以使用HTML 5或者XHTML结构，一个无序的UL列表，关键是该列表必须放在nav标签内部，导航列表代码如下：

```
<ul>
        <li class="selected"><a href="#">博客</a></li>
        <li><a href="#">关于</a></li>
        <li><a href="#">联系</a></li>
        <li class="subscribe"><a href="#">RSS</a></li>
</ul>
```

（3）介绍区域使用了section标签，并添加了名为intro的样式类，同时还在内部添加header标签，用于表示介绍区域的个体标题，这里可以看出header的用法，除了在整个文档中使用之外，还能在局部的section区域中使用，介绍区代码如下：

```
<section class="intro">
        <header>
                <h2>什么是HTML 5</h2>
        </header>
        <p>HTML 5是用于取代1999年所制定的HTML 4.01和XHTML 1.0标准的HTML标准版
        本，现在仍处于发展阶段，但大部分浏览器已经支持某些HTML 5技术。</p>
</section>
```

（4）主内容区同样使用了section标签，由于内容显示的是博客的文本，所以在内部添加一层article标签的嵌套，代码如下：

```
<section class="mainContent">
        <article class="blogPost">
                <header>
                        <h2>HTML 5新标签</h2>
                        <p>
                                时间<time datetime="2013-7-1">2013-7-1</time>
                                作者 <a href="#">小周</a>
                        </p>
                </header>
                <p>通过制定如何处理所有HTML元素以及如何从错误中恢复的精确规则，
```

```
        HTML 5改进了互操作性，并减少了开发成本。HTML 5还包含了新的元素，
        比如：<nav>, <header>, <footer> 以及 <figure> 等等。</p>
        </article>
</section>
```

（5）aside标签通常定义主体之外的内容，但是又与附近内容相关，一般会用于填充一些不同分类的内容，本例使用aside标签制作侧边栏，放置文章分类的标签，代码如下：

```
<aside>
        <section>
                <header>                        <-- 标题 -->
                        <h3>标签</h3>
                </header>
                <ul>
                <li><a href="#">HTML 5</a></li>  <-- 分类内容 -->
                </ul>
        </section>
</aside>
```

（6）最后一块内容是footer标签所表示的页脚，通常出现在section或者document的尾部，可包含一些与站点或者内容相关的信息，如作者信息、站点导航、友情链接等，本例所用的footer代码如下：

```
<footer>                                         <-- 页面页脚 -->
        <div>
                <section id="about">             <-- 关于 -->
                        <header>
                                <h3>关于</h3>
                        </header>
                        <p>关于内容</p>
                </section>
                <section id="blogroll">          <-- 友情链接 -->
                        <header>
                                <h3>友情链接</h3>
                        </header>
                        <ul>
                                <li><a href="#">大众点评网</a></li>
                        </ul>
                </section>
                <section id="popular">           <-- 相关流行内容 -->
                        <header>
                                <h3>流行</h3>
                        </header>
                        <ul>
                                <li><a href="#">流行内容</a></li>
                        </ul>
                </section>
        </div>
</footer>
```

HTML 5的到来使得浏览器更容易实现复杂布局，开发人员可以使用更少的时间去实现

以前的功能，同时可以将节约下来的时间去学习一直被前端开发者忽略的事情，如前端性能优化、前端架构等。

1.3.1 搭建开发HTML 5的浏览器环境

HTML 5是传统HTML的扩展和延伸，所以与传统的开发方式基本相同，笔者在Web开发时，通常有三件工具不可或缺：舒服的编辑器、强大的浏览器和代理工具。

1. 舒服的编辑器

在编辑器的选择上，Web前端开发自由度是非常高的，即使是文本文档编辑器也可以作为Web开发的工具，但是为了提高开发效率，还是要选择一款功能强大且流行的编辑器。笔者推荐的是近年来席卷前端界的Sublime Text，一款独具个性的高级编辑器，如图1.24是Sublime Text编辑器的截图。

```
Demonstration - Sublime Text 2
File Edit Selection Find View Goto Tools Project Preferences Help
base64.cc

void base64_encode(const uint8_t * data, size_t len, char * dst)
{
    size_t src_idx = 0;
    size_t dst_idx = 0;
    for (; (src_idx + 2) < len; src_idx += 3, dst_idx += 4)
    {
        uint8_t s0 = data[src_idx];
        uint8_t s1 = data[src_idx + 1];
        uint8_t s2 = data[src_idx + 2];

        dst[dst_idx + 0] = charset[(s0 & 0xfc) >> 2];
        dst[dst_idx + 1] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
        dst[dst_idx + 2] = charset[((s1 & 0x0f) << 2) | (s2 & 0xc0) >> 6];
        dst[dst_idx + 3] = charset[(s2 & 0x3f)];
    }

    if (src_idx < len)
    {
        uint8_t s0 = data[src_idx];
        uint8_t s1 = (src_idx + 1 < len) ? data[src_idx + 1] : 0;

        dst[dst_idx++] = charset[(s0 & 0xfc) >> 2];
        dst[dst_idx++] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
        if (src_idx + 1 < len)
            dst[dst_idx++] = charset[((s1 & 0x0f) << 2)];
    }

Line 31, Column 52                Spaces: 4        C++
```

图1.24 Sublime Text编辑器

Sublime Text支持目前主流的操作系统，如Windows、Mac、Linux，包括32位和64位，同时支持各种流行编程语言的语法高亮、代码补全等。该款编辑器插件相当丰富，同时版本更新快。非常棒的一点是编辑器右边没有滚动条，取而代之的是代码缩略图。Sublime Text是款收费软件，不过目前为止可以无限期的使用。



Sublime Text还有更多意想不到的强大功能，读者可以自行下载体验，编辑器下载地址为http://www.sublimetext.com/3。

2. 强大的浏览器

开发调试所使用的浏览器，笔者一般选择Google Chrome，这个在前面已经提过，选择

的原因是该款浏览器出色的脚本执行速度和页面渲染能力，同时还集成了强大的开发调试工具，如图1.25为该浏览器开发者模式的截图。

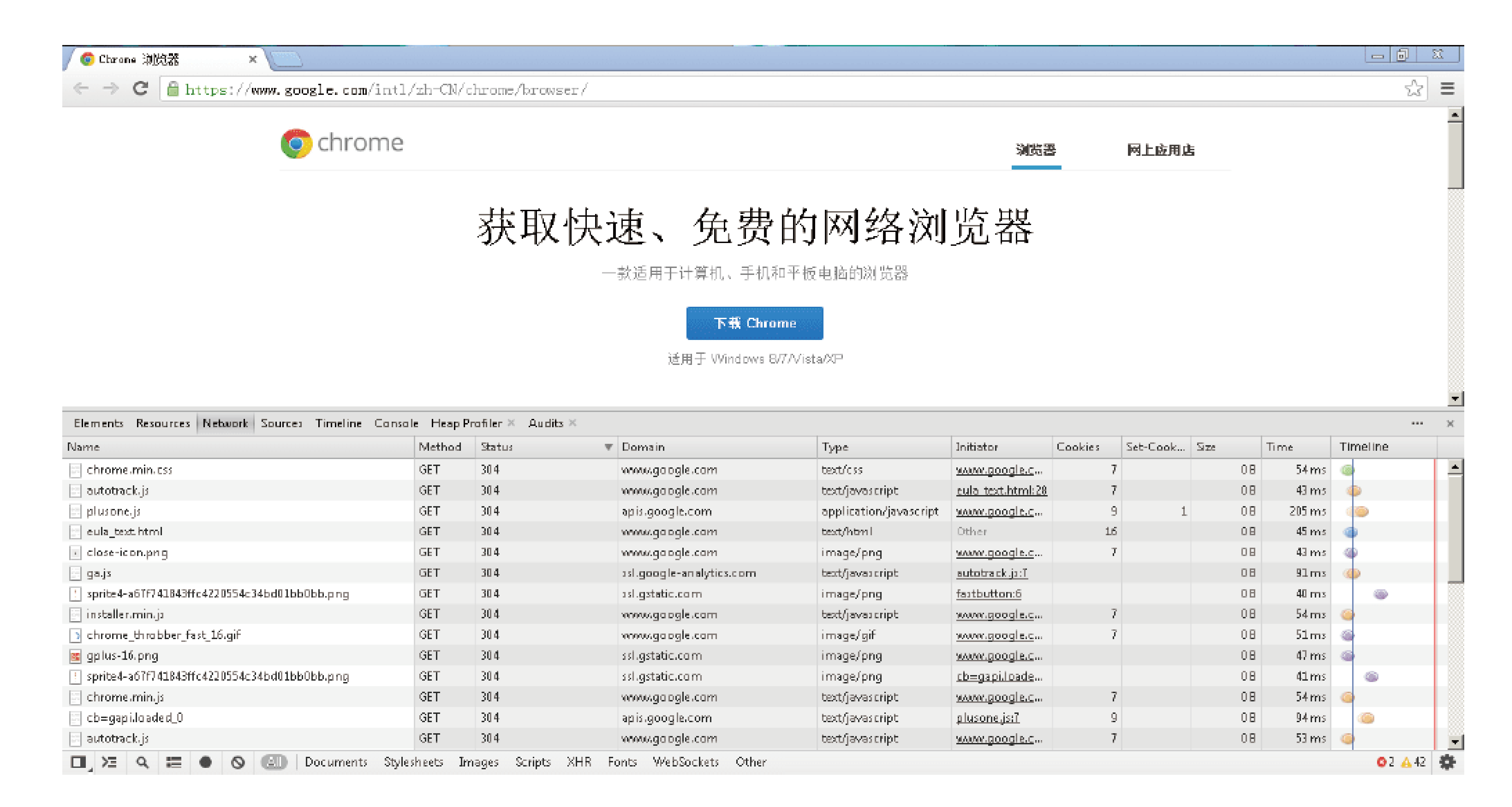


图1.25 Google Chrome 浏览器开发模式

3. 代理工具

第三样工具是请求代理工具，这里将介绍Windows操作系统上的Fiddler软件，如图1.26所示。

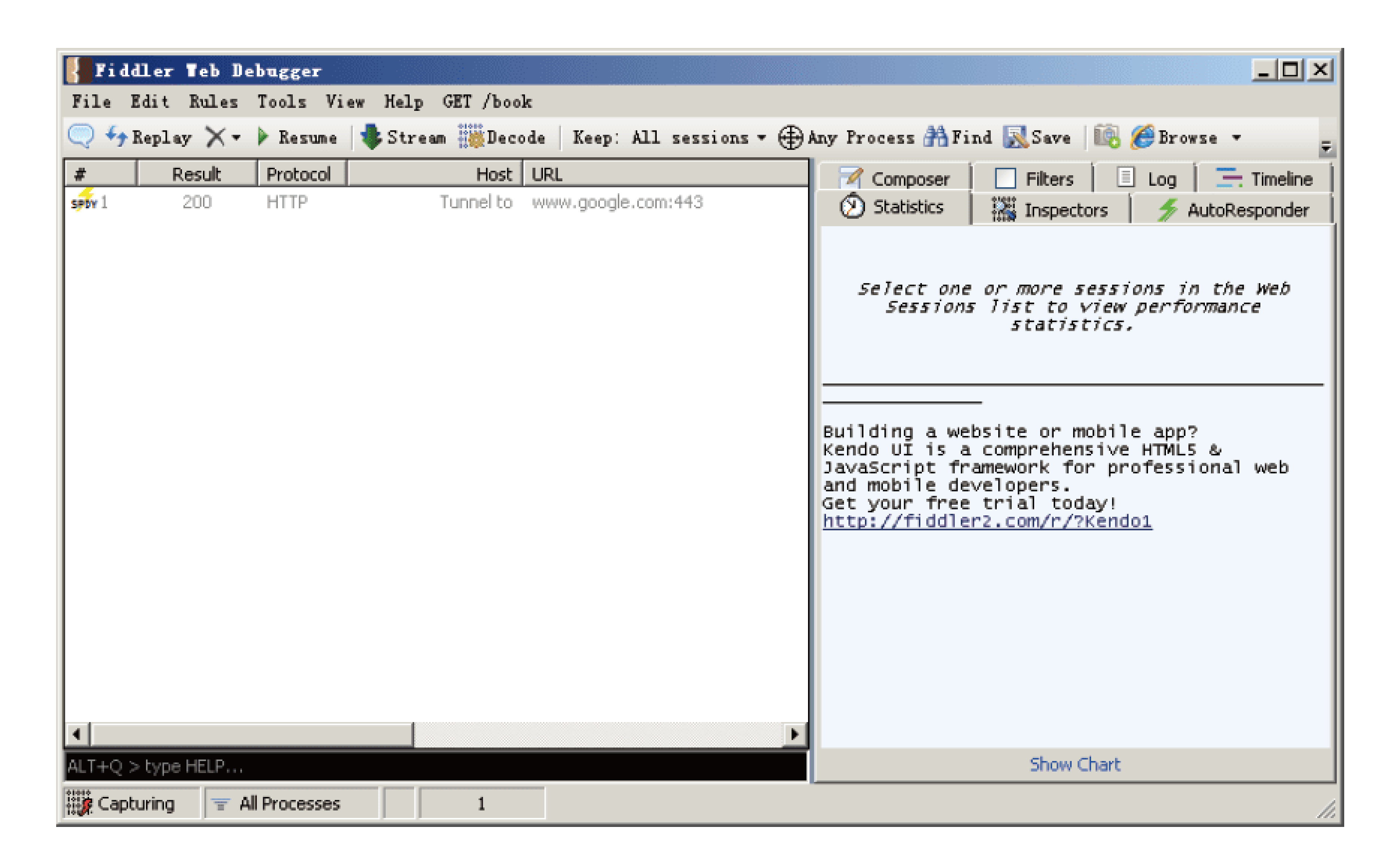


图1.26 Fiddler软件

虽然Fiddler主要的功能是数据包抓取，不过开发时常常用于请求代理，如图1.27为Fiddler代理的原理图。



图1.27 Fiddler代理的原理题

接着，举一个代理使用的例子，需求是将大众点评网首页代理到谷歌搜索首页，即将网址http://www.dianping.com代理到http://www.google.com网址上，Fiddler配置如图1.28所示。

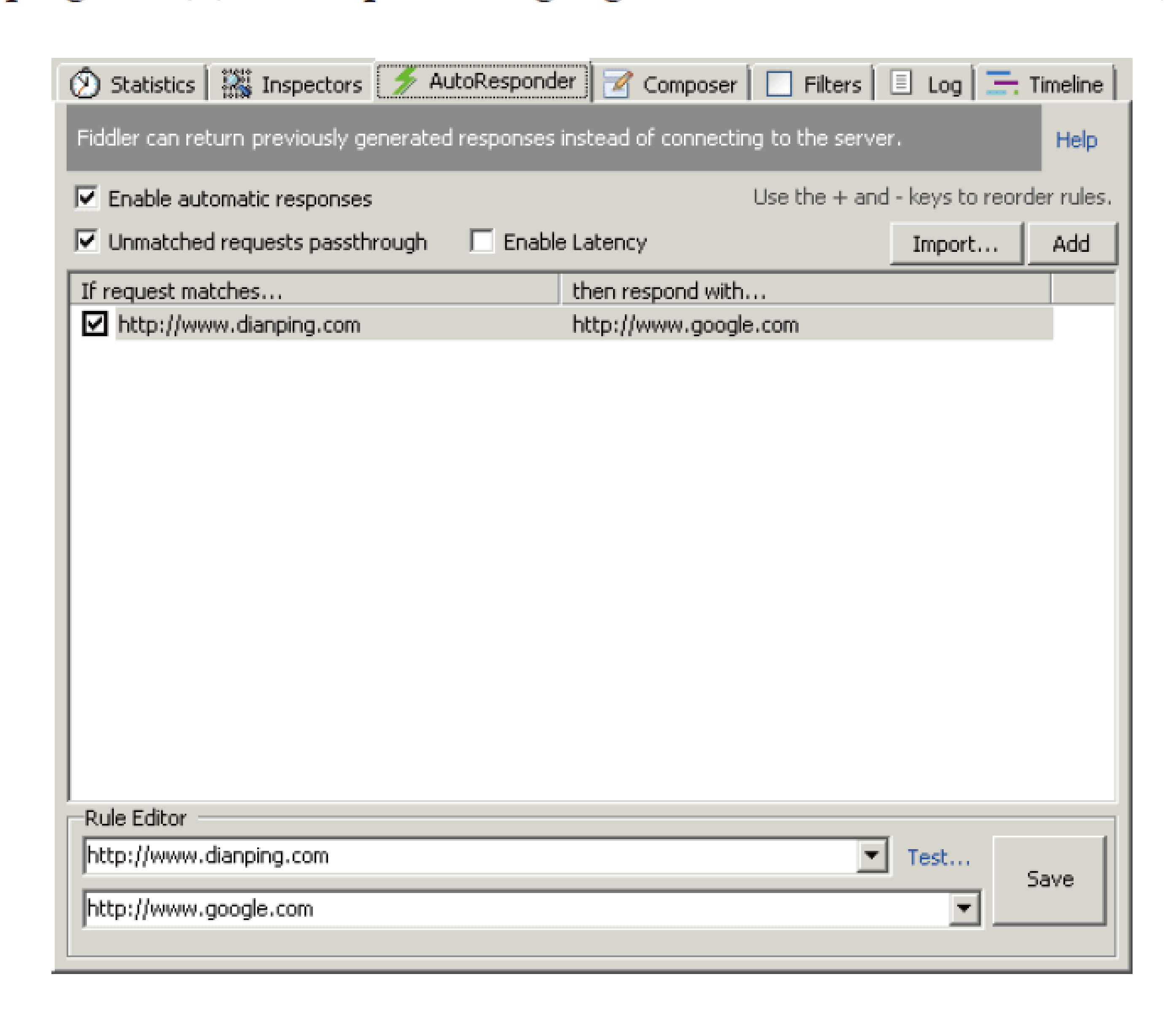


图1.28 Fiddler代理配置

在通常的开发情况下，一般不会将整个页面地址代理到另外一个地址上，而是代理一些样式表或者脚本用于本地调试开发。

最基本的开发环境就介绍到这里。Web开发还有很多优秀的工具，可以提高开发速度和测试效率，比如性能测试方面的DynaTrace Ajax Edition、JavaScript规范工具JSLint、IE测试工具IETest等等，读者可以根据开发习惯和喜好选择合适的工具。

1.3.2 检测浏览器是否支持HTML 5标签

在开发HTML 5应用时，为了实现浏览器的兼容性，需要对一些不支持新特性的浏览器做优雅降级，目的是为了尽可能使用户体验更加顺畅，这时候就需要对不同的浏览器做功能检测，通常有几种方法可以检测是否支持HTML 5标签，如原生的标签兼容提示、浏览器检测、特征检测等。

1. 原生的标签兼容提示

首先看原生标签提示，当浏览器遇到一些无法识别的标签时，只会将其作为一个普通文本节点进行展现，如HTML 5中的canvas标签，下面通过一个简单的实例对比支持与不支持浏

览器的区别，示例代码如下：

```
<!doctype html>
<html>
    <title>Canvas</title>
    <body>
        <canvas style="background-color:red">该浏览器不支持canvas</canvas>
    </body>
</html>
```

先用支持canvas标签的Google Chrome浏览器打开该文件，可以看到该标签能够正常显示，效果如图1.29所示。再用不支持canvas标签的IE 8打开文件，页面上显示“该浏览器不支持canvas”提示信息，效果如图1.30所示。

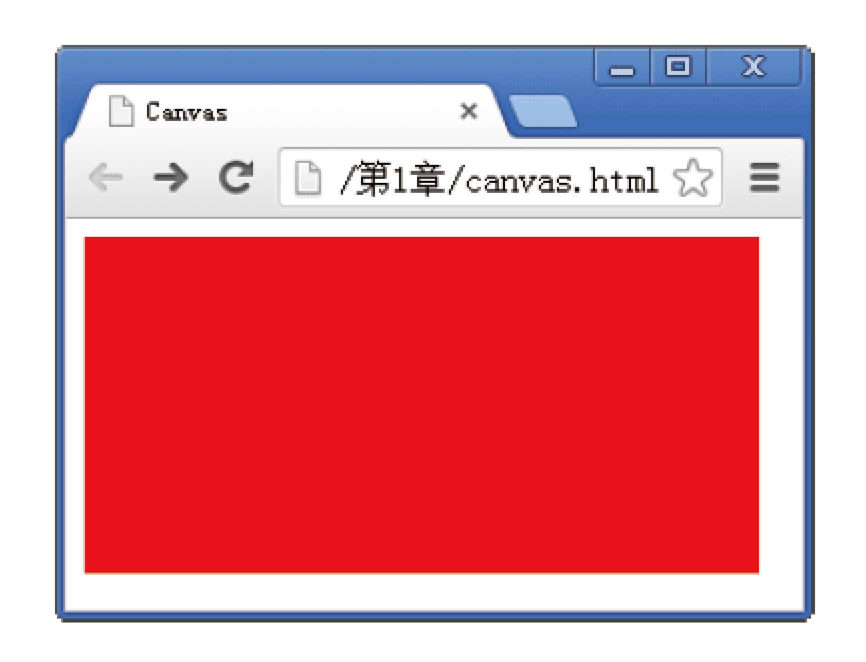


图1.29 Google Chrome浏览器打开网页文件

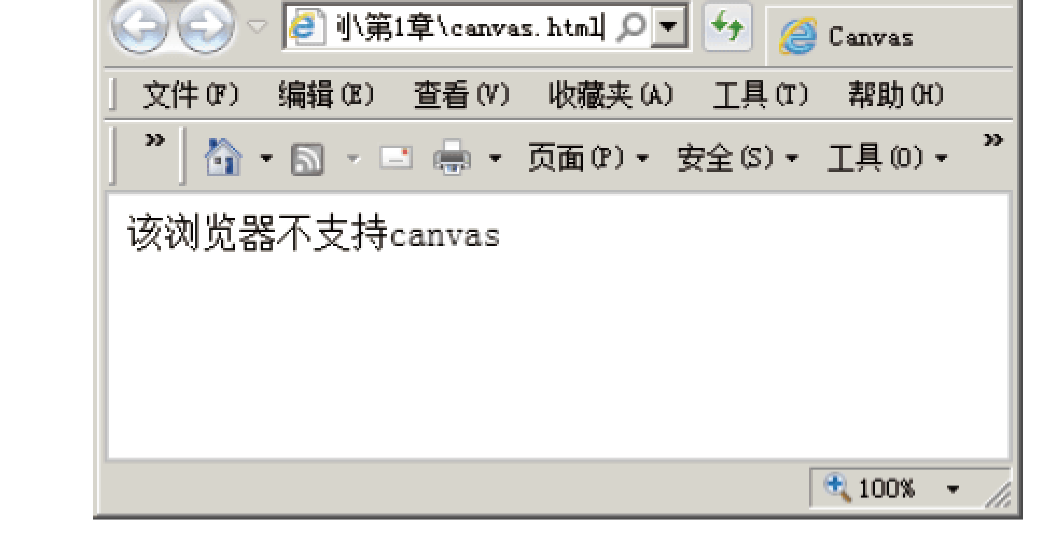


图1.30 IE 8浏览器打开网页文件

2．浏览器检测

另外一种常用的检测手段是用浏览器检测，主要原理是：先判断浏览器的类型和版本，然后在知晓该类型版本浏览器支持的情况下进行操作，下面通过一个示例代码介绍如何实现：

```
<!doctype html>
<html>
    <title>浏览器检测</title>
    <body><canvas style="background-color:red"></canvas></body>
     /* canvas标签 */
    <script>
        var ua = navigator.userAgent.toLowerCase(),ie;
        /* 获取浏览器User Agent */
        if (ie = ua.match(/msie ([\d.]+)/)) { /* 使用正则获取IE相关信息 */
            ie = ie[1] >> 0;                 /* 向右位移2位转为整型 */
            if (ie < 9) {                    /* 是否是IE9以下版本 */
                document.body.innerHTML = '该浏览器不支持canvas';
        }; };
</script></html>
```

该示例运行效果与上一个示例效果相同。示例开始先获取浏览器的User Agent，User

Agent是HTTP协议中头信息的一部分，用来告诉服务器客户端是什么浏览器，以及操作系统等的信息。通过分析User Agent获取浏览器的类型和版本信息，对不支持的浏览器采取错误提示。用浏览器检测这个方案看上去相当完美，不过有几个致命的缺点：

- User Agent可以伪造，很多浏览器支持用户修改User Agent信息。
- 因不断更新的浏览器版本，需要同步更新检测信息。

3. 特征检测

最后介绍的一种方案应该是最完美的，一般称为特征检测，即通过判断特定的对象、属性、方法、行为确认功能是否可以使用，下面通过示例介绍该方法的使用，代码如下：

```
01 <!doctype html>
02 <html>
03     <title>特征检测</title>
04     <body></body>
05     <script>
06         var _canvas = document.createElement('canvas');
07         if (_canvas.getContext) {
08             document.body.innerHTML = '<canvas style="background-
               color:red"></canvas>';
09         } else {
10             document.body.innerHTML = '该浏览器不支持canvas';
11         };
12     </script>
13 </html>
```

该段示例运行效果与首段示例“原生标签提示”效果相同。

代码第6行创建一个canvas标签元素，然后在第7行通过获取新创建canvas元素的getContex属性判断是否存在，当存在该属性时表示浏览器支持canvas元素。特征检测的优势在于，开发者不需要了解浏览器的支持情况，而是通过直接检测特性是否支持。



常见问题

从HTML 5问世以来，开发者就一直期待新带来的特性能解决以往Web开发中不能逾越的问题，事实上HTML 5确实解决了开发者很多现实问题，但同时也存在着部分暂时无法实现的功能，如：

- 本地存储数据安全性；
- 性能问题；
- 离线存储同步问题；
- 直播视频。

不过笔者相信，上述的一些问题在不久的将来都将被克服，HTML 5将会出现在一个更大的舞台上。

1.4.1 学好HTML 5要先学好Java吗

Web开发设计的知识点和技术比较庞杂，很多刚入门的新手对各种知识点的认识和把握不到位，容易引起误会，首先看两种技术的官方解释。

- HTML 5：是用于取代1999年所制定的HTML 4.01和XHTML 1.0标准的HTML标准版本，用于强化Web网页的表现性能，包括HTML、CSS和JavaScript在内的一套技术组合。
- Java：是个简单、面向对象、分布式、解释性、健壮、安全与系统无关、可移植、高性能、多线程和动态的语言。Sun Microsystems公司于1995年5月推出，由Java编程语言、Java类文件格式、Java虚拟机和Java应用程序接口4个方面组成。

通过概念可以看出，学习HTML 5与学习Java没有必要的关联关系。但作为开发人员，通过学习后端开发语言Java，有助于加深对前端开发的认识，如面向对象、设计模式的应用等，同时还能知晓完整的Web开发周期所经历的步骤和过程，所以表面上看技术之间千差万别，但又密不可分。除此之外，前端人员还可以补充如数据库、服务器开发方面的知识，对日后的日常开发会起到事半功倍的效果。如图1.31给出了前端开发的知识体系。

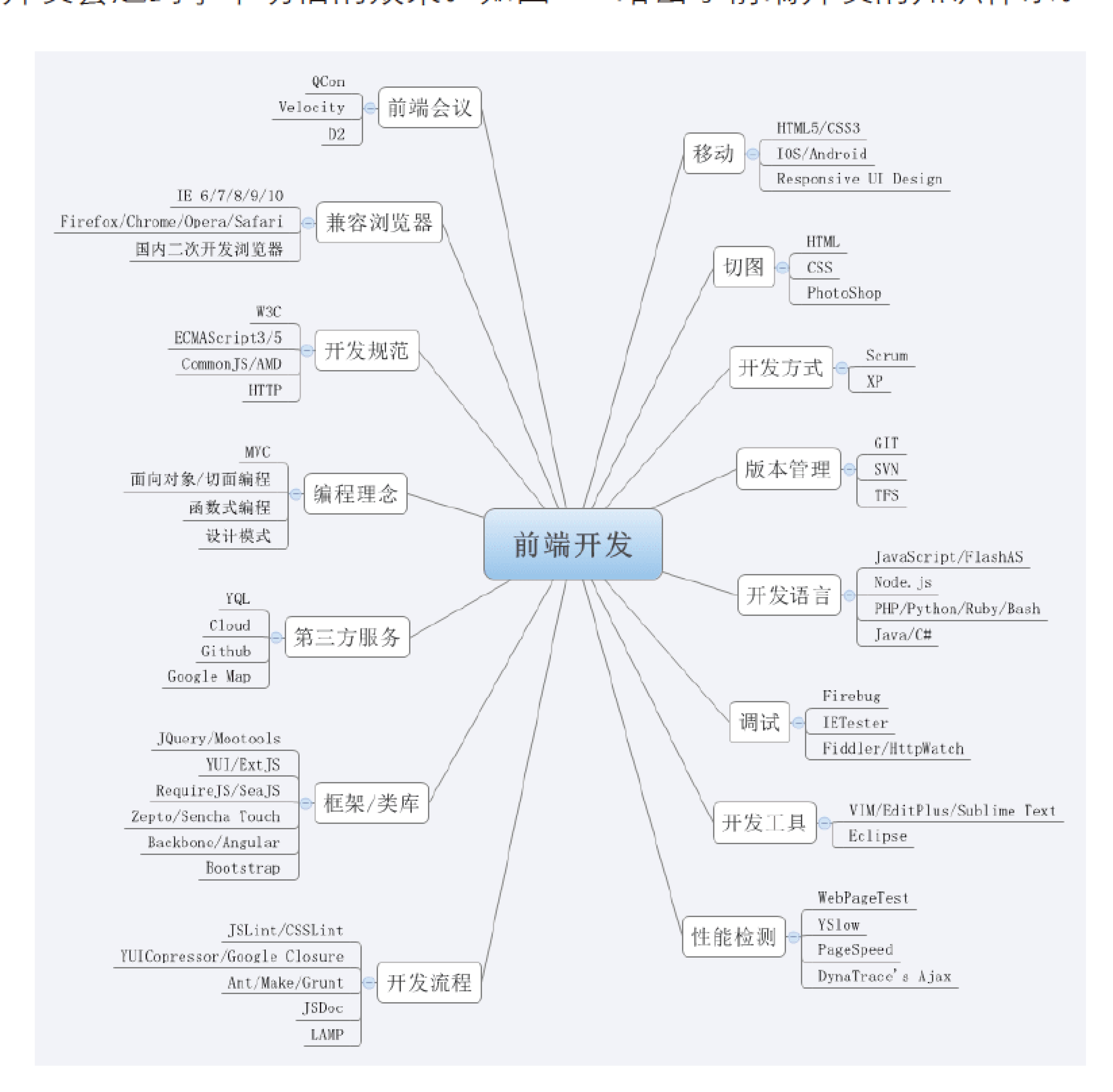


图1.31　前端开发知识体系题

1.4.2 谁是HTML 5新规则下的牺牲品

从2006年开始，富互联网应用进入如火如荼的白热化战争，各种富应用技术百家争鸣，如Adobe Flex、JavaFX、SilverLight等，一时之间让开发者不知如何选择。各种常见的富应用如下。

- Adobe Flex：由Macromedia公司在2004年3月发布，基于其专有的Macromedia Flash平台，支持富应用开发和部署的技术组合。
- JavaFX：由Sun公司开发的一种声明性的、静态类型脚本语言，开发富互联网应用程序。
- SilverLight：融合了微软多种技术的Web呈现技术，提供了一套开发框架，并通过使用基于向量的图像图层技术，支持任何尺寸图像的无缝整合，对基于ASP.NET、Ajax在内的Web开发环境实现了无缝连接。

学习任何一门技术都将投入大量的时间和精力，包括笔者在内的多数Web开发人员，急切需要一种类似于HTML的升级方案。终于，HTML 5的第一份正式草案于2008年1月22日公布，虽然还不是很完善，且浏览器支持情况不同，但还是让人看到了希望。以JavaScript、CSS、HTML为基础的三剑客，似乎又将再续辉煌。

与其他的技术相比，HTML 5有自己的优势，如：

- 用户可以不用再安装和更新浏览器上的各种插件。
- 即时的更新，不需要升级安装，只需要刷新网页。
- 跨域设备和平台。
- 公开的标准，由W3C推出。
- 良好的SEO支持，更容易被搜索引擎收纳。

2013年，越来越多的网站开发使用一部分HTML 5技术，相信未来HTML 5会给大家带来更大的想象空间。

1.4.3 HTML 5是否有未来

从HTML 5诞生这日起，笔者就一直认为它的前途将一片光明，由HTML 5作为HTML的一种补足更加的自然、顺畅。

近些年来，业内涌现了一大批使用HTML 5开发的应用，涵盖了视频、游戏等各方面，下面通过笔者搜寻的一些例子向大家展现HTML 5取得的成就。

游戏Cut The Rope（割绳子），苹果iOS/Android平台上一款非常受欢迎的休闲游戏，如图1.32是该款游戏的网页版，使用HTML 5编写，支持Google Chrome、Safari、Firefox、IE 9+等。



图1.32 HTML 5版“割绳子”游戏界面

网页版没有触控支持，其他效果与手机上基本没有区别，尤其在Google Chrome浏览器上表现非常出色，游戏网址为http://www.cuttherope.ie/。

再推荐一款HTML 5应用Google Body，展现的是一个3D人体模型，用户可以剥离各个解剖层，放大并定位至感兴趣的部位，单击即可获悉具体解剖结构，或者搜索肌肉、器官、骨骼及更多构造了解详情。（浏览器需支持WebGL），网址为http://www.zygotebody.com/。图1.33为Google Body应用的截图。

提示

WebGL是一种3D绘图标准，这种绘图技术标准允许把JavaScript和OpenGL ES 2.0结合在一起，通过增加OpenGL ES 2.0的一个JavaScript绑定，WebGL可以为HTML 5 Canvas提供硬件3D加速渲染，这样Web开发人员就可以借助系统显卡来在浏览器里更流畅地展示3D场景和模型了，还能创建复杂的导航和数据视觉化（摘自百度百科）。



图1.33 Google Body应用界面

目前，市面上的HTML 5应用非常丰富，但有很大一部分的应用处于演示阶段，不过在可以预见的将来HTML 5有无限的可能性，原本单一的浏览器环境正在慢慢变得多元化。同时，大量高级HTML 5游戏引擎出现，更加丰富了原有Web开发人员的多元性，越来越多开发者把目光放在HTML 5身上，可以很肯定地说，HTML 5王朝正在崛起，新的帝国正在崛起。

1.4.4 HTML 5在移动应用开发是否有前景

2012年的9月12日，对于HTML 5开发者是一个莫大的打击，Facebook CEO扎克伯格参加了TechCrunch Disrupt大会，接受了阿灵顿（Michael Arrington）的采访，指出可能犯的最大错误就是“在HTML 5上下赌注太多，在本地程序下注太少。”，之后，美国第二大社交网站LinkedIn也宣布放弃HTML 5，而改用原生代码来实现，总结起来原因大致如下：

- 应用占内存过大。
- 没有原生应用性能好，如旋转、滚动效果。
- 周边配套调试工具少，优化困难。

在国内，情况更加复杂，网络访问速度成为更大的障碍。虽然，HTML 5有离线缓存，同时很多大型公司也在优化HTTP协议或者开发更快的新协议，但就目前来说HTML 5离移动商业化还有一段距离。但是，在HTML 5的身上笔者看到了当年JavaScript的影子。早期的JavaScript被用在表单验证，后来泛滥为制作漂浮广告等一些微不足道的地方，但俗话说是金子总会发光，现在的JavaScript已然是浏览器端的霸主，截止目前JavaScript已经是排行前十的语言，如图1.34为2013年1月份的编程语言排行榜。

1	2	↑	C
2	1	↓	Java
3	5	↑↑	Objective-C
4	4	=	C++
5	3	↓↓	C#
6	6	=	PHP
7	7	=	(Visual) Basic
8	8	=	Python
9	9	=	Perl
10	10	=	JavaScript

图1.34 2013年1月编程语言排行

移动基于原生开发与HTML 5如同当年的PC本地应用之与浏览器。计算机诞生之初，网络还没有广泛普及，当时的人们使用软盘或者光盘的方式进行大数据传输。随后，互联网的普及与浏览器的出现，打破了操作系统本地应用的神话，越来越多的本地应用被搬上浏览器，浏览器更像是一个即时更新的操作系统。设想一下未来操作系统的样子，在网络速度足够快的环境，所有的数据都将存储在云端，可以通过任何形式的终端共享数据，平日的应用都可以通过浏览器完成，操作系统将变得越来越轻量级，不管在何时、何地都能畅快地进行分享、办公，也许这个梦想一点也不远，Firefox OS手机已经面市，Chrome Book也在售卖，大型的公司正在开发更多在线应用，这样的未来还会远吗？

当然，现在的HTML 5还存在诸多的问题，大家对它也是褒贬不一，但是，依托于HTML的广阔应用，HTML 5作为其功能的补足和扩展，完全有理由相信HTML 5的一部分成熟功能，将会得到充分的利用。对于年轻一代的开发者来说，HTML 5必然是首选技能。



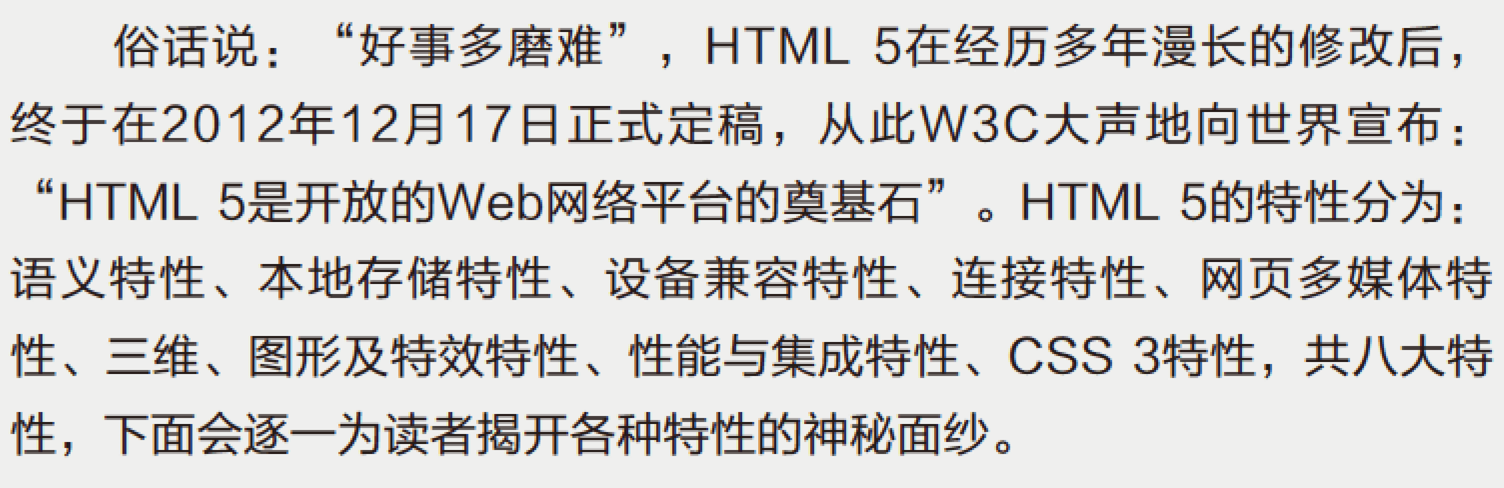
1.5 本章小结

本章主要讲述了HTML 5的发展史，介绍各个HTML版本的演变，并给出HTML 5特性结构图。接着，通过不同浏览器的比较，了解HTML 5的支持情况和兼容方案，同时通过示例展现HTML 5引入的新标签，感受HTML 5带来的简洁特性。本章同时还介绍了作为一位前端工程师所要具备的技能，帮助大家更好的学习前端知识，最后，解答了学习HTML 5的常见问题，从技术和终端平台方向帮助大家理清思路，确定目标。

从下一章开始，将具体介绍HTML 5的各种新特性。



第 2 章 HTML 5的整体特性



俗话说：“好事多磨难”，HTML 5在经历多年漫长的修改后，终于在2012年12月17日正式定稿，从此W3C大声地向世界宣布：“HTML 5是开放的Web网络平台的奠基石”。HTML 5的特性分为：语义特性、本地存储特性、设备兼容特性、连接特性、网页多媒体特性、三维、图形及特效特性、性能与集成特性、CSS 3特性，共八大特性，下面会逐一为读者揭开各种特性的神秘面纱。

本章知识点：

- HTML 5的新元素
- Web存储与离线Web应用
- 音频视频与地理位置
- HTML 5的通信技术
- CSS 3技术



2.1 HTML 5的新元素

在HTML 5新元素出现之前，编写HTML一直试图用DIV去模拟各种形态，如section（章节）、header（页眉）、footer（页脚）、nav（导航）、article（条目）等，通过元素的ID属性赋予DIV新的特征，零碎的命名和没有统一的规范使得页面语义差强人意，对页面的SEO（Search Engine Optimization，搜索引擎优化）缺乏友善地引导，HTML 5在元素上的追加和增强弥补了这种不足。设计师只需要按照设定好的元素标签名称就能设计出在语义上近乎完美的页面，HTML 5在语义上的完善，使其在“编写一次，随处运行”的理想上迈出了坚实的一步，不论使用何种设备，都能和谐地达到统一。

虽然目前不是所有浏览器都能完美地支持HTML 5的新元素，但由于浏览器在设计之初对于无法识别的元素会进行忽略，所以使用者可以放心地在开发的项目中使用HTML 5带来的新元素。或许，目前看来一切还不是那么地完美，但就像浏览器刚刚诞生之初的不尽如人意，随着网络的飞速发展，HTML 5将会展开命运的翅膀，飞向光明的未来。

2.1.1 最新的交互元素——内容交互、菜单交互、状态交互

HTML 5不仅仅只带来了一堆在语义上进行强化了的元素，同时在交互效果上也进行了

极大地改变。也就是说，使用者可以在不使用JavaScript的情况下也能制作出满足一般需求的交互效果，这在笔者看来是一个非常友善的突破，可以通过HTML结构读出更多统一的使用规范和动态语义的交互信息。

首先看显示内容上的交互特性，这里向读者介绍的是details与summary两个元素。details元素用来描述文档或文档片段的信息，summary元素与details元素一同使用，用于说明文档的标题，同时，summary元素应为details元素的第一个子元素。通过一个简单的示例展示summary与details的使用，代码如下：

```
<!DOCTYPE HTML>
<html>
<style>details details{padding:15px}</style>
<!-- 二级details样式-->
<body>
<details>
  <summary>交互</summary>
  <!-- 第一级detauls标题-->
  <details>
    <summary>内容交互</summary>
    <!-- 第二级detauls标题-->
    <p>details与summary元素</p>
    <!-- 文档内容 -->
  </details>
</details></body></html>
```

details与summary元素示例的交互效果如图2.1所示。

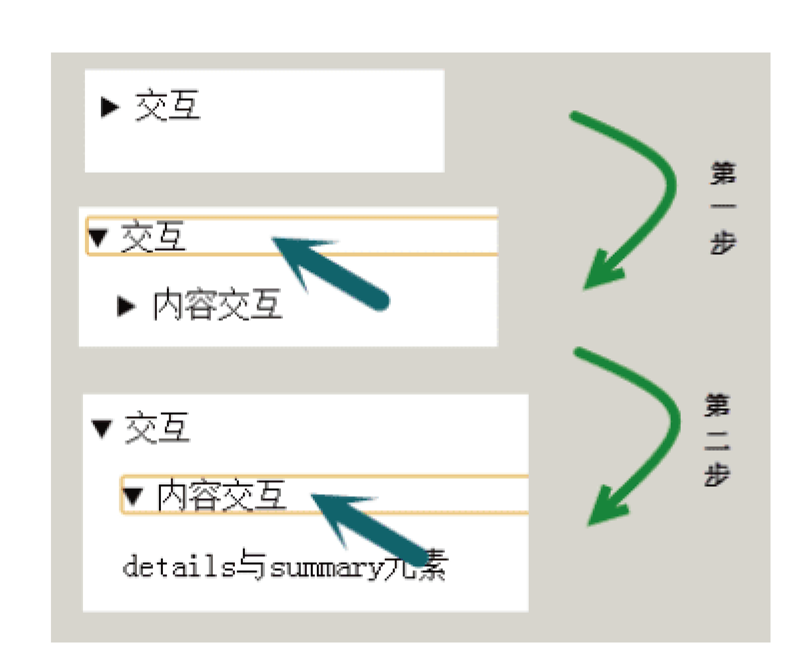


图2.1　details与summary元素示例

details同时还具备open属性，用于表示是否展开可见，语法如下：

```
<details open="open">
```

在菜单交互中，命运最坎坷的应属menu元素，这个早在HTML 2时代就出现的元素，居然在HTML 4时遭到弃用，幸运的是HTML 5从语义的角度重新拾回menu元素。menu元素出现时常与li元素一同使用，用于排列表单控件。另外一个是command元素，按照W3C的说明，该元素有单选按钮、复选框、按钮三种类型，目前笔者测试了市面上主流的浏览器，都暂时无法使用command元素特性，读者如果有兴趣可以自行尝试。

最后一类是状态交互类型，即可以理解为页面上的进度条。如progress元素、meter元

素。progress元素一般用于下载或者上传时的进度显示，当状态产生变化时，可以通过设置progress的属性改变元素的交互状态。progress元素有两个关键属性。

- value：进度的当前值，可以为整数或者浮点数。
- max：完成的值，即总量，可以为整数或者浮点数。

下面通过一个简单的示例展示progress元素的使用，代码如下：

```
<!DOCTYPE HTML>
<html>
<body>
      下载进度：<progress value=30 max=100></progress>
      <!-- 当前进度-->
</body>
</html>
```

运行效果如图2.2所示。

下载进度：

图2.2 progress元素使用效果

meter元素与progress元素非常相似，主要用于定义度量衡，表示在一定范围内的值，如投票占比、使用量等。meter元素有6个关键属性。

- value：实际度量的值，默认为0，可以为整数或者浮点数。
- high：范围的上限值。
- low：范围的下限值。
- max：最大值，默认值是1。
- min：最小值，默认值是0。
- optimum：最佳的值，必须在min属性值与max属性值之间。

下面通过一个示例说明meter元素的使用，代码如下：

```
<!DOCTYPE html>
<html>
<body>
      <p>投票结果：</p>
      <p>小周：                                    <!-- 小周的投票值 -->
      <meter value="20" optimum="100" high="100" low="0" max="100"
      min="0"></meter><span>20%</span>
      </p>
      <p>小王：                                    <!-- 小王的投票值 -->
      <meter value="80" optimum="100" high="100" low="0" max="100"
      min="0"></meter><span>80%</span>
      </p>
</body>
</html>
```

meter元素示例的运行效果如图2.3所示。

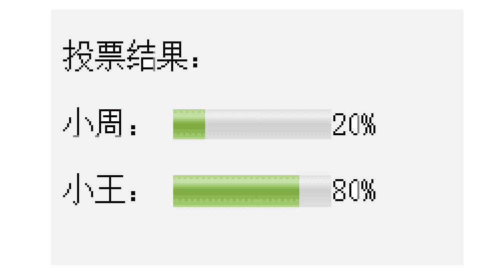


图2.3　使用meter元素效果

2.1.2 HTML 5页面结构

HTML 5在结构上散发着简洁之美，同时又不失良好的语义结构。先看传统的HTML 5出现之前的网页设计结构，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- 文档规范 -->
<html xmlns="http://www.w3.org/1999/xhtml">      <!-- XML命名空间 -->
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <!-- 页面编码 -->
    <meta name="keywords" content="xxx" />          <!--关键字 -->
    <meta name="description" content="xxx" />      <!--本页描述或关键字描述 -->
    <title>标题</title>                              <!-- 页面标题 -->
</head>
    <body>内容</body>
</html>
```

如果使用HTML 5诠释上面的页面，代码如下：

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="keywords" content="关键字" />
    <meta name="description" content="此网页描述" />
    <title>标题</title>
</head>
    <body>内容</body>
</html>
```

对比两段示例代码可以看到，不用再去复制冗长的文档规范，而且不用在意doctype是大写还是小写，这是极简的经典体现。另外在html元素上的xmlns属性也得到了释放，这个早期用于设定XML命名空间的方法可以退出历史舞台了。



meta属性中的keywords和description类型，主要用于提供搜索引擎优化查询，让用户在使用搜索引擎时能快速地通过关键字查找到对应页面。

在HTML 5的页面中经常可以看到增强语义性的元素，通过元素的字面英文就能大致理解元素的使用场景，下面通过代码示例向读者介绍常用元素的使用场景和使用方法。

第一个要介绍的是header元素，用于显示页面或者指定区域的页眉或头部，使用方法如下：

```
<header>
    <h1>HTML 5页面</h1>
    <p>header元素示例</p>
</header>
```

hgroup元素用于对网页或区段的标题进行组合，如对文章的标题和副标题进行组合，使用方法如下：

```
<hgroup>
    <h1>HTML 5元素</h1>
    <h2>hgroup元素</h2>
</hgroup>
<p>标题类的组合</p>
```

aside元素用于定义所处内容之外的内容，但又与附近内容有相关性，如用于显示某篇文章的作者信息，使用场景代码如下：

```
<p>HTML 5文章内容,略....</p>
<aside>
    <h1>作者介绍</h1>
    <p>小周，资深Web前端技术开发。</p>
</aside>
```

section元素用于定义文档中的节，如章节、页眉、页脚或文档中的其他部分，如下代码表示文档中的段落：

```
<section>
    <h1>HTML 5</h1>
    <p>HTML 5是用于取代1999年所制定的HTML 4.01和XHTML 1.0标准的HTML标准版本...</p>
</section>
```

article元素用于定义独立的内容，如来自论坛帖子、新闻报纸的文章、博客文本、网站用户评论等，如下面代码表示一篇文章：

```
<article>
    <header>
          <hgroup>
                <h1>HTML 5网页案例大全</h1>
                <h2> HTML 5的整体特性</h2>
          </hgroup
    </header>
    <p> HTML 5页面结构内容</p>
</article>
```

HTML 5为使用者带来了大量的语义元素标签，表面上看会带来大量的学习成本，但一旦熟练使用，编写一张语义清晰并且对搜索引擎友好的页面将是轻而易举的事情。如果想成为一位合格的前端开发者，那么学习这些新标签是学习HTML 5的第一步也是最关键的一步。

2.1.3 DOCTYPE和字符集

在2.1.2节中通过新老DOCTYPE的对比，读者可以清晰地看到HTML 5在精简旧有结构上做出的努力。DOCTYPE在出现之初主要用于XML中，用作描述XML允许使用的元素、属性

和排列方式。起初HTML借鉴了XML中DOCTYPE的使用方法，并赋予了新用法，如大家熟知的触发浏览器的标准模式。假使在制作一张页面时，没有设定DOCTYPE，则浏览器会以怪异模式状态进行处理（即Quirks模式），该模式与标准模式在模型、样式、布局等都存在较大差异。因此，DOCTYPE在制作页面时是不可或缺的部分。

在HTML 4标准中，DOCTYPE被分为三种模式：

- 严格模式，即严格遵循W3C标准的模式，代码格式如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">
```

- 过渡模式，包含了W3C标准的内容，同时还可以使用不被W3C推荐的标签，如font、b等，而且不可以使用框架元素（即frameset元素），代码格式如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
ww.w3.org/TR/html4/loose.dtd">
```

- 框架模式，可以使用框架元素，其他与过渡模式相同，代码格式如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.
w3.org/TR/html4/frameset.dtd">
```



之前的HTML关于DOCTYPE标准，读者可以参看W3C官方文档说明，网页地址为http://www.w3.org/TR/1999/REC-html401-19991224/struct/global.html#h-7.2。

虽然之前版本的HTML对DOCTYPE做了重重规定，但真实情况却是浏览器会尽最大的可能渲染对应的页面，即使可能出现了一些不符合模式的做法，唯一会出现的是浏览器会在控制台中显示一些错误警告，这种做法就是常说的浏览器容错性，实则是对市场和用户的妥协。

发展到HTML 5，W3C将剔除DOCTYPE原有鸡肋的声明方式，简化为如下格式：

```
<!DOCTYPE html>
```

对于绝大多数开发者来说，只需要使用这一种方式就足以满足日常的开发使用，但如果要考虑到日后的兼容和扩展等一系列问题，还需要了解W3C在新制定DOCTYPE的一些新规定，主要分为三类：

- 普通模式，即<!DOCTYPE html>
- 弃用模式，听起来不知所云，其实指的就是对过往模式的兼容模式，不过HTML 5弃用了之前的过渡模式和框架模式，最终留下了6种书写格式，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN">
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/
REC-html40/strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/
xhtml11/DTD/xhtml11.dtd">
```

- 遗留兼容模式，对于过往无法考证规则的一种兼容方式，语法格式如下：

```
<!doctype HTML system "about:legacy-compat">
```

通过对浏览器DOCTYPE的理解，读者可以熟悉浏览器模式的触发方式，不过就通常开发而言，只需要使用<!DOCTYPE html>这一种普通模式。



HTML 5三种最新模式可以查看链接http://dev.w3.org/html5/markup/syntax.html#doctype-syntax。

所谓的字符是对各种文字和符号的总称，涵盖了各国文字、标点符号、图形符号和数字等。字符集是对多个字符的集合，常用的字符集有ASCII、GB2312、Unicode、ISO等。科学家为了让计算机准确地处理各种字符集，需要对字符进行编码，以便计算机能够识别和存储各种文字。

在HTML 5出现之前，浏览器会根据三种方式确认页面的编码格式，按优先级排列如下：

- 获取HTTP请求头中的Content-Type字符对应的值。
- 使用meta标签声明，语法格式如下：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

- 外链资源使用charset属性声明编码格式，如在script标签中使用语法格式如下：

```
<script type="text/javascript" src="myscripts.js" charset="UTF-8">
</script>
```

HTML 5出现后，对字符集的使用做了大量的简化，可以使用以下语法进行字符集声明：

```
<meta charset="utf-8">
```

对于日常使用网站开发而言，结合HTML 5的字符集使用，笔者给出如下建议：

- 最优先使用HTTP请求头指定编码。
- 统一全站字符集编码，HTML 5推荐UTF-8字符集。
- 使用meta标签确认字符集编码，尽可能放在html标签的第一个子元素位置。
- 第三方引用的脚本，在不确认字符编码时，加上charset属性设置编码格式。

2.1.4 其他标签元素

HTML 5通过良好的语义格式，引入了大量新的标签，除了之前介绍的header、hgroup、aside、section、article等以外，本节还将继续为读者介绍其他标签元素。

datalist元素，与input元素配合使用，定义input可能出现的值，下面通过一个示例说明datalist元素的使用，代码如下：

```
<!DOCTYPE HTML>
<html>
<body>
<input list="province" />                              // 省份输入框
<datalist id="province">                               // 省份数据
      <option value="北京">                            // 省份数据元素“北京”
      <option value="上海">                            // 省份数据元素“上海”
```

```
        <option value="浙江">                         // 省份数据元素“浙江”
</datalist>
</body>
</html>
```

当用户在文本框内输入“北”字，则输入框下方自动出现补全提示，效果如图2.4所示。

北
北京

图2.4 datalist元素

文本框使用datalist的数据进行提示时，将input的list属性设置为指定datalist的id属性，使用起来非常简单。

mark元素，用于标记文本，突出显示文本内容，类似于文本高亮的作用，使用方法如下：

```
<!DOCTYPE HTML>
<html>
<body>
<p><mark>HTML5</mark>是用于取代1999年所制定的 HTML 4.01 和 XHTML 1.0 标准的 HTML
标准版本，现在仍处于发展阶段，但大部分浏览器已经支持某些<mark>HTML5</mark>技术。</p>
</body>
</html>
```

显示的页面中，“HTML5”部分因为mark元素的包围被高亮显示，效果如图2.5所示。

HTML5是用于取代1999年所制定的 HTML 4.01 和 XHTML 1.0 标准的
HTML 标准版本，现在仍处于发展阶段，但大部分浏览器已经支持某
些HTML5技术。

图2.5 mark元素高亮关键字“HTML5”

video元素，用于显示视频，如开发者要在自己的网站上添加一段演示视频，使用代码如下：

```
<video src="movie.ogg" controls="controls">
您的浏览器不支持video标签。
</video>
```

video元素作为一个相对复杂的使用元素，因此具备较多控制属性，如表2.1所示。

表2.1 video元素属性

属性	说明
autoplay	视频加载就绪后自动播放
controls	是否显示video默认控件，如前进、停止、声音控制等
height	播放器高度，以像素计
width	播放器宽度，以像素计
loop	循环播放
preload	页面加载时自动进行视频加载
src	播放视频的URL

有视频标签，一定少不了音频标签，HTML 5提供了audio元素用于播放音频信息，使用代码如下：

```
<audio src="audio.wav">
您的浏览器不支持audio标签。
</audio>
```

audio元素具备与video元素相同的控制属性，同样，当浏览器不支持audio时，会在页面上显示元素中间的文本提示信息。

HTML 5还有众多新元素，刚刚介绍的还只是冰山一角，其他元素在本节就不再一一介绍了，希望读者在学习HTML 5新元素的同时，能够领悟HTML 5设计者的良苦用心。



2.2 Modernizr库

作为一名前端工程师，早已经习惯了处理多种浏览器的兼容性。同样，如果开发者想使用HTML 5也必然要对一些老的浏览器进行兼容和支持，对于一些无法兼容的特性做出适当的提示。一般较为常见的安全检测手段有4种，如下：

- 检查全局对象，如window、navigator是否拥有指定属性，如离线存储、地址位置信息等HTML 5新特性。
- 通过创建新元素，检查元素对象上是否拥有指定的HTML 5属性，如Canvas等。
- 通过创建新元素，检查元素对象上是否拥有指定的HTML 5方法，同时调用该方法，并判断返回值，如检查video元素支持的视频格式。
- 通过创建新元素，设置元素对象上的HTML 5指定属性值，并判断设定后的值是否被保留。

可见，检测HTML 5特性需要熟知各种情况，这并非是一件简单的事。幸运的是目前市面上已经出现了一些第三方库用于检测HTML 5的特性，下面将介绍目前最热门的Modernizr库。

2.2.1 Modernizr库是什么

Modernizr是一个用JavaScript编写的开源类库，用于检测浏览器是否支持CSS 3和HTML 5的新功能。Modernizr库比传统的通过浏览器的UserAgent获取浏览器版本信息判断特性支持的方法更可靠，检测结束后将判断结果存储在Modernizr对象上，使用者可以通过获取Modernizr对象上的信息判断浏览器对CSS 3或者HTML 5的支持情况。



传统的通过UserAgent判断浏览器，并通过对应浏览器版本号判断是否支持指定特性的方法的缺点在于要不断新增的浏览器和版本号，使得要对UserAgent的判断进行同步更新，同时检测浏览器版本号兼容的CSS 3或者HTML 5也是一项巨大而且非常有风险的任务。

使用Modernizr类库可以前往官网下载，地址为http://modernizr.com/，页面上提供了

development（未压缩版本，包含注释）和production（压缩版本）两种版本，如图2.6所示。



图2.6 Modernizr官网

进入development下载页面，可以发现Modernizr提供了各类属性检测的选择项，使得开发者可以按照自身的需求下载应用包的最小集，如图2.7所示。

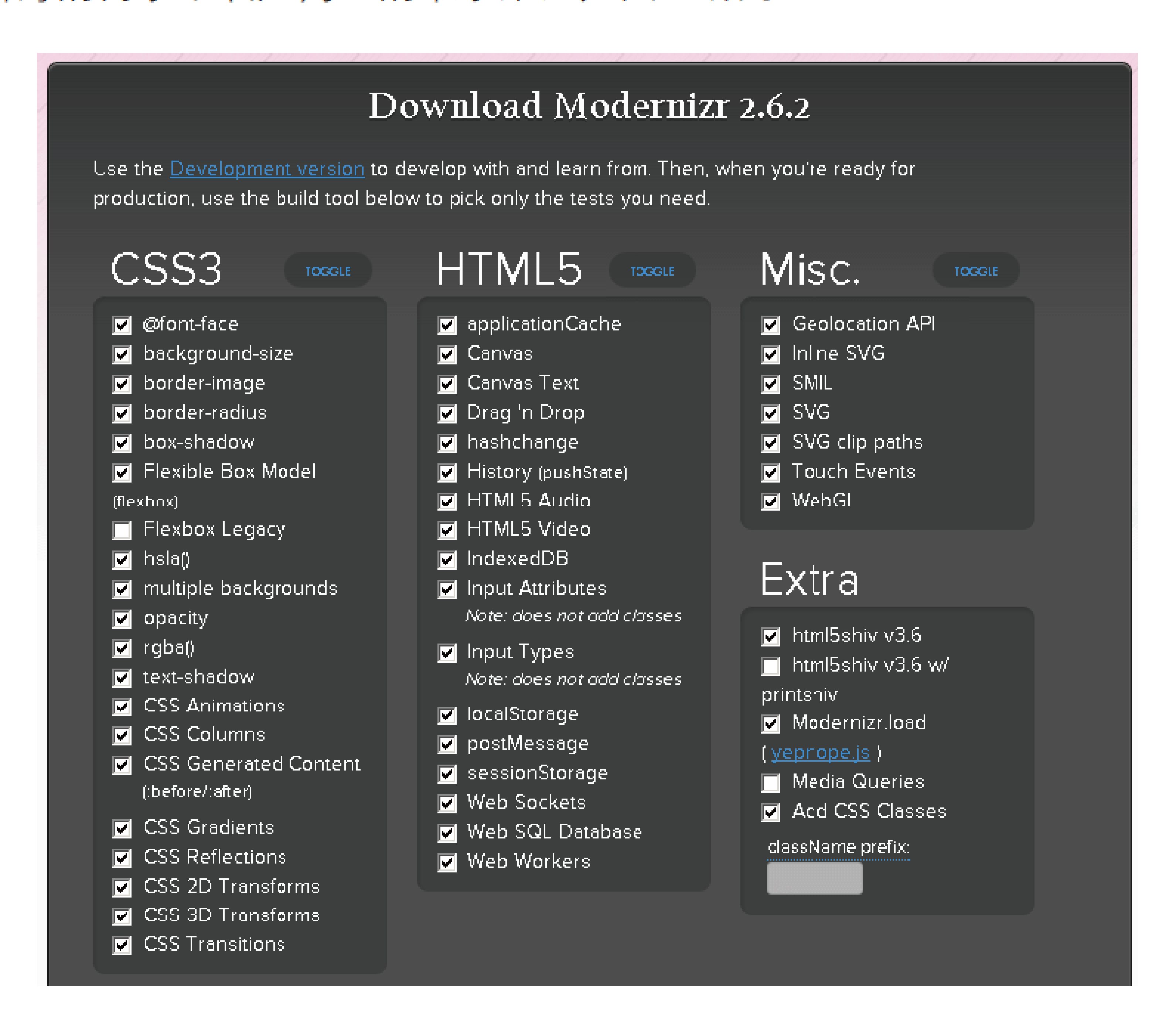


图2.7 Modernizr类库检测包选择

2.2.2 使用Modernizr库提供的方法检测浏览器的各项指标

本节将通过多个示例向读者展示Modernizr的部分使用方法。同一般的外链脚本一样，首先在使用页面引入Modernizr脚本，示例代码如下：

```
<!DOCTYPE html>
<html>
<head> <script src="modernizr.js"></script></head>
<!-- 引入Modernizr脚本 -->
</html>
```

当页面加载完毕后，Modernizr类库会将检测过的属性以class类名的方式添加到html元素上，如果上例的代码在浏览器运行处理后的页面转换成HTML，代码如下：

```
<!DOCTYPE html>
<html class=" js flexbox canvas canvastext webgl no-touch geolocation
postmessage websqldatabase indexeddb hashchange history draganddrop
websockets rgba hsla multiplebgs backgroundsize borderimage borderradius
boxshadow textshadow opacity cssanimations csscolumns cssgradients
cssreflections csstransforms csstransforms3d csstransitions fontface
generatedcontent video audio localstorage sessionstorage webworkers
applicationcache svg inlinesvg smil svgclippaths" style=""><head>
<!-- 已class名显示支持与不支持的属性 -->
<script src="modernizr.js"></script>
</head>
<body></body></html>
```

通过处理后的页面代码可以看到，Modernizr将检测到的HTML 5属性名都列在了html元素的class类名上，同时对于不支持的属性会添加“no-”的前缀，可以看到其中no-touch就是按照这样的规则产生的。

Modernizr在页面初始完毕后，还会创建一个名为Modernizr的JavaScript对象，该对象的每个属性都是检测浏览器完毕以后对应HTML 5属性名的布尔值，如判断浏览器是否支持WebSockets，那么可以用如下代码进行检测：

```
if(Modernizr.websockets ){
    console.log('支持websockets');}else{
    console.log('不支持websockets');};
```

不仅可以通过JavaScript的方式进行检测，使用者还可以结合class类名进行操作，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
    .yes-tip, .no-tip{ display: none;}              <!-- 默认提示不显示 -->
    .canvas .yes-tip{color:green; display: block;}
    <!-- 支持canvas，显示提示为绿色 -->
    .no-canvas .no-tip{ color:red; display: block; }
    <!-- 不支持canvas，显示提示为红色 -->
</style>
<script src="modernizr.js"></script>
</head>
<body>
    <div class="yes-tip">您的浏览器支持canvas</div>    <!-- 绿色支持提示 -->
    <div class="no-tip">您的浏览器不支持canvas</div>    <!-- 红色不支持提示 -->
</body>
</html>
```

示例的原理非常简单，当加载Modernizr类库以后，会默认为浏览器的html元素添加支持HTML 5特性的class类名。当使用支持HTML 5 Canvas的浏览器打开页面时，html元素被添加名为canvas的样式类名至class属性，页面文档中样式类名为yes-tip的div元素被激活显示，同时出现绿色的“您的浏览器支持canvas”提示，反之亦然。

通常绝大部分浏览器使用audio的同时会实现ogg、mp3、wav、m4a这4种基本的视频格式，但不排除部分浏览器只实现了部分格式，下面可以通过Modernizr类库判断浏览器，并提供最佳的视频加载格式，代码如下：

```
var audio = new Audio();
audio.src = Modernizr.audio.ogg ? 'background.ogg' :        // 是否支持ogg格式视频
            Modernizr.audio.mp3 ? 'background.mp3' :     // 是否支持mp3格式
                                  'background.m4a';     // 是否支持m4a格式
audio.play();
```

代码首先会创建一个Audio对象用于播放视频，Modernizr执行完毕后，会将检测完毕的视频格式添加至Modernizr的audio对象属性上，示例的代码中先通过Modernizr.audio.ogg判断ogg格式视频，当浏览器支持时则Modernizr.audio.ogg的值为true，即加载名为background.ogg的视频文件，如果不支持则继续执行，其他视频格式判断方式与ogg格式视频相同。

虽然HTML 5的出现可以让开发者使用大量新的特性，但同时还是有很大一部分普通用户使用着不支持HTML 5的老浏览器访问网页，在无法迅速地让用户升级浏览器的情况下，这时候诞生了一种名为Polyfills的技术，即在老版本的浏览器中实现统一标准的应用程序接口。

比方说，开发的某个应用要实现获取用户的地理位置经纬度信息，需要借助HTML 5的Geolocation功能，但对于老版本的浏览器该功能无法实现，这时候就需要通过第三方的服务去模拟Geolocation，如谷歌提供的地理位置服务。为了让业务层的代码更加清晰，让普通开发者不需要关心浏览器层面的兼容问题，这时需要实现一个与HTML 5的Geolocation应用程序接口相同的代码包，Modernizr在这方面也提供了对应的加载支持，示例代码如下：

```
Modernizr.load({
    test: Modernizr.geolocation,              // 判断条件
    yep : 'geo.js',                           // 支持geolocation加载的脚本
    nope: 'geo-polyfill.js'                   // 不支持geolocation加载的脚本
});
```

在GitHub上的Modernizr库中，其开发团队为大家搜集了与HTML 5相关的Polyfills，地址为https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills，如果无法访问请使用代理。



GitHub是目前最火的基于Git的代码托管平台，并提供了一个Web界面和各种系统的可视化操作工具，是管理软件开发和发掘学习代码的首选。

Modernizr的使用远远不止这些，同时该类库还在开发团队的努力下不断演化，如果读者想了解更多的开发使用信息，可以前往官网的开发者文档，地址为http://modernizr.com/docs/。



2.3 表单和文件

表单应用占据了网页功能的半壁江山，说到表单，最先让人联想到的是表单元素，一个可以容纳文本输入、下拉列表、单选项、复选框等等的超级元素。传统的表单使用有诸多限制，比如表单元素只会提交表单标签内部的元素集合，表单操作行为必须写在表单自身的属性上。HTML 5在这方面的改进取得了长足的进步，初次接触到HTML 5中的表单时，一定会被这大胆的设计所折服。

同样，文件操作一直是HTML 5出现之前网页功能开发挥之不去的软肋，通常的做法是需要借助第三方的工具读取和操作文件，如使用Flash等。如此的开发，增加了开发的复杂度和不确定性，大大提高了开发和维护成本，因此HTML 5提出了一个全新的File对象用于设置和获取文件信息。接下来，将会为读者详细介绍表单和文件的使用，一同感受HTML 5全新的设计思想。

2.3.1 input元素的新增类型

表单中使用最多的莫过于input元素，主要用于填写用户信息，传统的input元素有10种类型，如表2.2所示。

表2.2 传统的input元素类型

类型名称	说明
button	可单击的按钮
checkbox	复选框
file	输入字段和“浏览器”按钮，用于文件上传
hidden	隐藏的输入字段
image	图像形式的提交按钮
password	密码输入框，输入的字符被掩码
radio	单选按钮
reset	重置按钮，清空表单内的所有数据
submit	提交按钮，提交表单数据至远程服务器
text	文本输入框

HTML 5在此基础上做了进一步地加强，添加了另外13种类型，如表2.3所示。

表2.3 HTML 5新增input元素类型

类型名称	说明
date	带日历控件的日期字段
datetime	带日历和时间控件的日期字段
datetime-local	带日历和事件控件的日期字段
email	电子邮箱文本字段
month	带日历控件的日期字段的月份
number	带数字控件的数字字段
range	带滑动控件的数字字段
time	带时间控件的日期字段的时、分、秒

（续表）

类型名称	说明
url	URL文本字段
week	带日历控件的日期字段“周”
color	拾色器
search	用于搜索的文本字段
tel	用于电话号码的文本字段

为了便于读者观察学习，图2.8呈现了各种input元素新类型在网页中的预览效果。

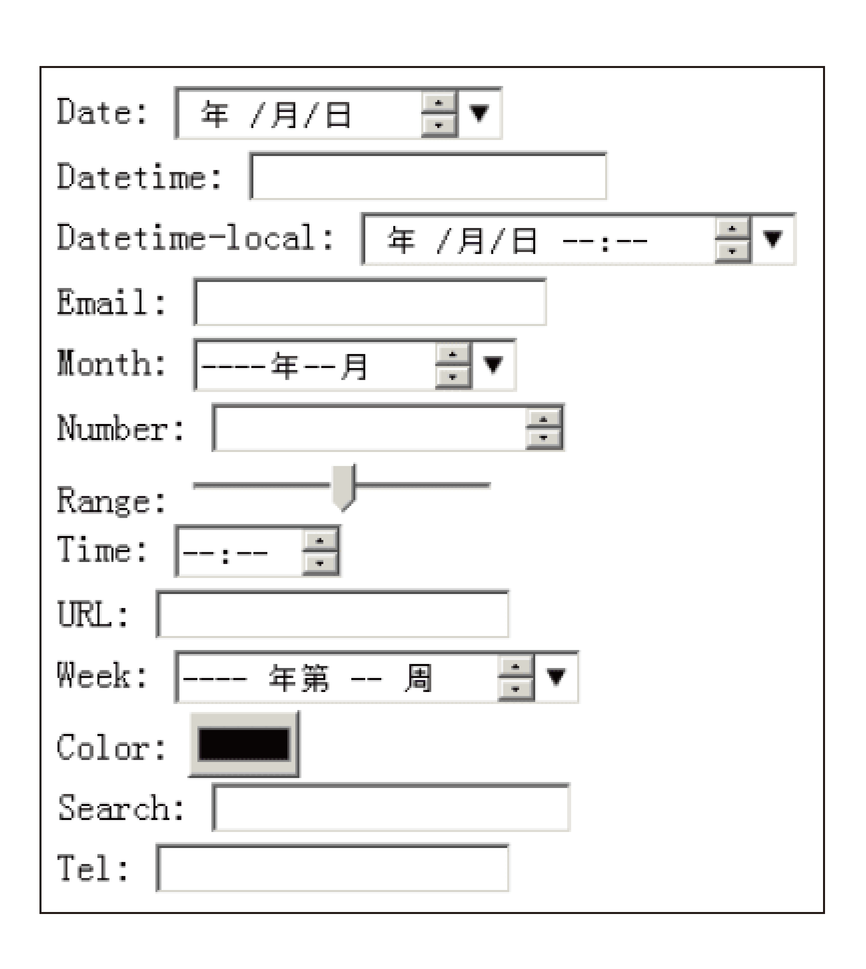


图2.8 HTML 5新增input元素类型的预览效果



目前不是所有浏览器都支持上述input新类型，读者可以使用Chrome最新版进行浏览。

2.3.2 input元素新增的公用属性

HTML除了带来令人激动的input新类型，同时还带来了非常多的input新属性。以往需要用JavaScript实现的功能，现在只需要修改一个属性就能搞定。HTML 5新标准带来了更加细致的分工，使得开发者拥有更多便捷的使用选择。

下面将介绍input新属性，首先通过一个表格看看到底都新增了哪些属性，如表2.4所示。

表2.4 HTML 5新增input元素属性

属性名称	说明
autocomplete	使用输入自动完成功能
autofocus	页面在载入成功后获得焦点，type类型为hidden的除外
form	规定字段所属表单，可为多个
formaction	覆盖所属表单的action属性，适用于type类型为submit、image
formenctype	覆盖所属表单的enctype属性，适用于type类型为submit、image
formethod	覆盖所属表单的method属性，适用于type类型为submit、image
formnovalidate	覆盖所属表单的novalidate属性，适用于type类型为submit、image
formtarget	覆盖所属表单的target属性，适用于type类型为submit、image
list	引用输入字段提示的对应datalist数据

（续表）

属性名称	说明
max	输入数字的最大值
min	输入数字的最小值
pattern	输入字段值的模式，正则格式
placeholder	输入字段提示
required	表示输入字段的值是必需的，不为空
step	输入数字的间隔
multiple	允许选择多个上传文件

接下来将结合实际使用场景介绍部分新增属性功能，使读者能够在平时的开发中得以灵活地运用。

1. autocomplete属性

autocomplete属性默认是开启状态，浏览器自身会记录页面对应文本框每次输入的信息，当用户键入相关的字符时，会出现相关的提示信息。但有些情况下用户并不希望浏览器自己做出提示，因为很多时候用户错误的输入信息也会被记录下来。在很多第三方类库中都会提供下拉提示插件，即当用户输入文字时，将与文字相关的信息在输入文本框下方进行提示选择，使用者可以非常方便地在输入部分关键字后找到想要的结果，如百度搜索框的下拉提示，如图2.9所示。



图2.9 百度查询下拉提示



对于使用下拉提示效果的文本输入框，都会将autocomplete属性设置为off，以避免浏览器默认提示框与JavaScript插件模拟的提示框在页面上相互交叠。

2. autofocus属性

autofocus属性默认为关闭状态，在该属性没有出现之前，要实现类似的功能需要借助于JavaScript。一般input元素的autofocus属性会用在表单的第一个填写元素上，如常见的登录页面，效果如图2.10所示。



图2.10 登录页面

autofocus属性主要的用处是，当页面加载完毕后，用户可以通过键盘直接输入信息，而不需要通过鼠标点选输入框后再进行输入，是一种交互体验上的优化。为了在老版本的浏览器中实现类似于autofocus属性的效果，可以通过JavaScript强行触发元素的focus事件，代码如下：

```
document.getElementById('id').focus()
```



正常的情况下，一张页面只会出现一个被设置了autofocus属性的input元素。

3. form属性

form属性的出现，使得input元素从表单结构中被完全解放出来。使用过HTML 5之前表单的开发者会知道，只有将input元素放置在表单标签内，数据才能同表单一起被提交，这样的设计使得页面元素利用率大打折扣，页面设计上出现过许多无用的结构。新属性的使用方法如下：

```
<form action="login" method="post" id="login_form"><input type="submit"
/></form>
账号:<input type="text" name="name" form="login_form"/>
密码: <input type="password" name="password" form="login_form" />
```



当一个input元素被用于多个表单，可以在form属性上将各表单id值以空格符隔开填写。

4. 重写表单的input属性

HTML 5的input元素中，还有一类新属性用于重写input元素所属表单属性，如以下各项。

- formaction：重写表单的action属性，action用于设定发送表单数据的请求地址。
- formenctype：重写表单的enctype属性，enctype表示发送表单的编码格式，如multipart/form-data，表单默认以application/x-www-form-urlencoded编码格式进行发送。
- formmethod：重写表单的method属性，method表示发送表单数据的方法，常用的如get、post。

- formnovalidate：重写表单的novalidate属性，novalidate表示不对表单数据进行浏览器默认验证，如验证input类型为url、email、telephoned的元素。
- formtarget：重写表单的target属性，target属性规定在何处打开提交请求。

5. list属性

input元素的list属性是一个相当与时俱进的改进，将以往用JavaScript实现的输入选择框通过原生的浏览器功能实现，使用方法如下：

```
网站：<input type="url" list="url_list" name="link" />
<datalist id="url_list">
<option label="谷歌" value="http://www.google.com" />
<option label="百度" value="http://www.baidu.com" />
<option label="淘宝" value="http://www.taobao.com" />
<option label="大众点评网" value="http://www.dianping.com" />
</datalist>
```

示例效果如图2.11所示。



图2.11 input元素list使用

6. placeholder属性

input元素的placeholder属性是一类实战性非常强的应用实现。多数网站在设计输入框的同时，为了达到良好的交互体验，会在输入框的空白输入处添加一段灰色的提示文字，方便用户按照设计意图进行填写。在placeholder属性出现之前通常有两类实现方法：

- 监听input元素的focus和blur事件，在获取焦点时，将预先输入在文本框内的文字清空，并修改文本框的输入字体颜色。
- 在文本框的上方悬浮一个用于提示的label标签，当单击标签时，隐藏标签并通过脚本强行设置下方文本框获取焦点。

placeholder使用方法如下：

```
链接：<input type="url" name="url"  placeholder="http://www.dianping.com" />
```

placeholder属性的出现有效地解决了这类常用问题，相同的功能可以完全使用原生的属性进行替代，减少了开发代码量和开发成本。

HTML 5除了为input元素新增刚刚介绍的6种属性外，还有新增了之前表2.4中出现的其他属性功能。

2.3.3 新增表单元素

上一节介绍了表单中最常用的input元素在HTML 5中新的使用，本节将介绍其他表单元

素通过HTML 5强化后的效果。

1. output元素

output元素用于表单内各种类型的输入，在功能上并没有很大的新意，不过从HTML 5的语义角度出发，确实丰富了表单元素的多样性。output元素可以监听forminput事件，forminput事件用于监听表单内的输入，每当输入结束后触发。下面通过一个示例介绍output元素使用，代码如下：

```
<!DOCTYPE html>
<html>
<body>
<form>
      <label for=" age ">年龄：</label>
      <input type="range" name="age" min="1" max="100" step="1"/>
      <output onforminput="this.value=age.value">30</output>
</form>
</body>
</html>
```

示例演示了一个选择年龄输入框，同时在选择时自动在output元素中显示选中的数值，效果如图2.12所示。

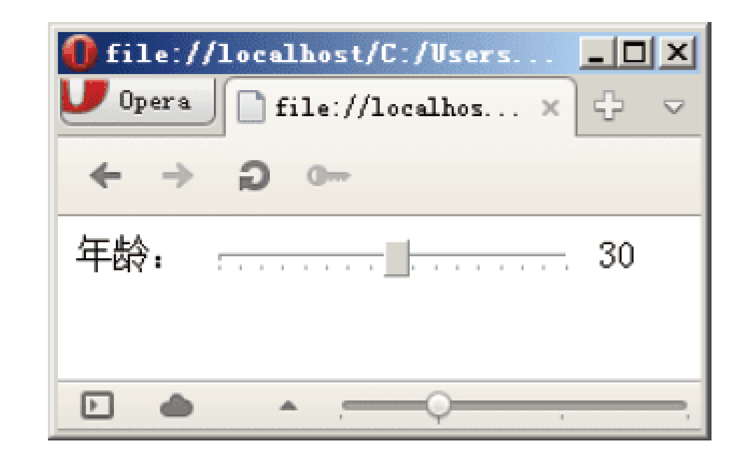


图2.12 output元素使用

用户可以拖动示例中的区域输入框，同时右侧的数字会显示最新的数值。在HTML 5出现之前，要实现类似的功能需要监测input中的数字是否发生变化，在变化时通过脚本改变输出的值，该方法的缺陷在于当表单中出现多个元素时，开发者需要监听每个输入元素的数值变化。HTML 5的output元素使得事情变得简单，使用时只需要监听output元素的forminput事情，当表单中的数据发生变化时会对forminput事件进行触发通知。



笔者使用Opera 12.14版本浏览器预览output元素示例。

2. keygen元素

keygen元素用于表单提交后的用户验证，当表单提交时，keygen元素会生成两个密钥：一个私钥存放在用户浏览器客户端，另外一个公钥被发送至服务器。目前，不同浏览器对keygen的支持程度不同，支持情况如图2.13所示。

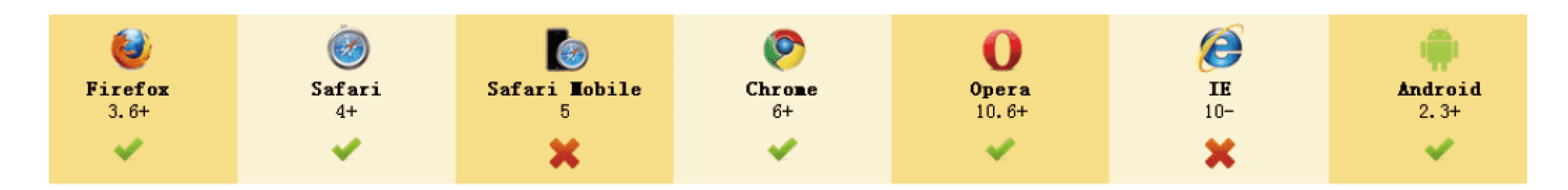


图2.13 keygen浏览器支持情况

不仅如此，浏览器支持的形式也不同，同样是key元素有不同的加密形式，如图2.14所示。

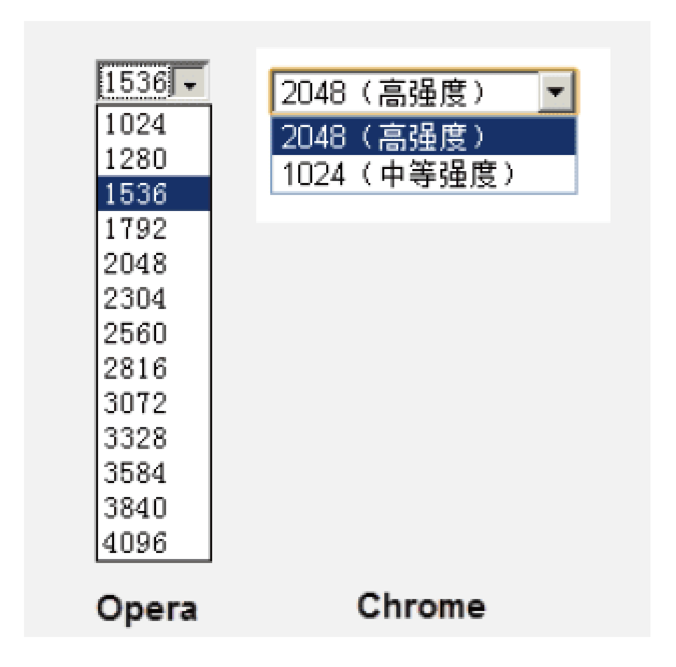


图2.14 Opera和Chrome的keygen元素对比

HTML 5的keygen元素属性如表2.5所示。

表2.5 keygen元素属性

属性名称	说明
challenge	属性中的值随着公钥一同提交至服务器
form	定义所属表单，一个或者多个
keytype	密钥类型，默认为RSA
autofocus	页面加载完毕后获得焦点

2.3.4 表单新增的验证方法

HTML 5除了给表单带来了多种类型的input元素和多种新元素外，同时还加强了表单验证方面的体验。以往开发者想要给表单添加验证有两种方法：

- 通过浏览器脚本监听表单提交事情，验证提交数据是否符合输入格式，并给出提示。
- 在表单提交后，后台服务器对表单数据进行验证，对于无法通过的数据重新输出页面进行提示。

HTML 5给开发者带来了第三种表单验证方法，通过表单元素内置的验证属性，当使用者提交表单，浏览器会根据各元素自身的验证属性进行判断并给出提示。下面将介绍与表单相关验证的属性。

1. novalidate属性

novalidate属性可以作用于form和input元素，表示该元素区域不进行表单验证。表单默认在提交时会验证类型为url、telephone、email、date的元素，代码如下：

```
<!DOCTYPE HTML>
<html><body>
<form method="get">
网址：<input type="url" name="user_email" />          <!-- 网址输入框 -->
<input type="submit" />                                <!-- 提交按钮 -->
</form>
</body></html>
```

网页运行效果如图2.15所示。

图2.15 带有url类型的input元素表单

在文本框内输入不符合URL规则的内容，并单击“提交”按钮，表单出现“请输入网址。”的信息提示，效果如图2.16所示。



图2.16 提交表单出现错误提示

假如在某些情况下，输入的网址内容并不一定需要严格遵循规范要求，则可以通过在form元素上添加novalidate属性解决，代码如下：

```
<form method="get" novalidate="true"></form>
```

2. pattern属性

pattern属性用于验证input提交内容的格式，验证格式为正则表达式。如下例显示只能输入三个数字的文本框，代码如下：

```
<!DOCTYPE HTML>
<html><body>
<form method="get">
3位数：<input type="text" name="num" pattern="[0-9]{3}" />
      <!-- 输入框 -->
<input type="submit" />
                         <!-- 提交按钮 -->
</form>
</body></html>
```

在文本输入框内输入a，并单击“提交”按钮，此时输入的内容与pattern属性的验证规则不相符，表单出现“请与所请求的格式保持一致。”的提示，效果如图2.17所示。

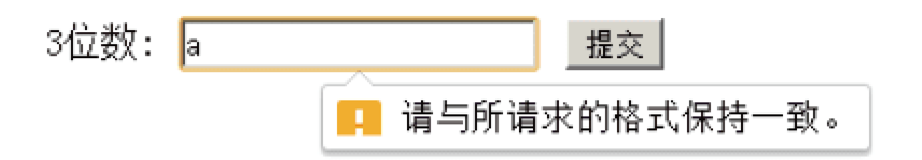


图2.17 pattern属性提交提示



正则表达式是用一个“字符串”来描述一个特征，然后去验证另一个“字符串”是否符合这个特征。比如表达式“ab+”描述的特征是“一个a和任意个b”，那么ab、abb、abbbbbbbbbb都符合这个特征。正则表达式的使用可以参考http://msdn.microsoft.com/zh-cn/library/ae5bf541(VS.80).aspx。

3. required属性

required属性作用于input元素，规定输入的内容不能为空，可以与之前的novalidate和pattern属性组合使用，示例代码如下：

```
<!DOCTYPE HTML>
<html><body>
<form method="get">
邮箱：<input type="email" name="email" required="required" />
            <!-- 邮箱输入框 -->
<input type="submit" />                          <!-- 表单提交按钮 -->
</form>
</body></html>
```

当用户不输入任何内容，直接单击“提交”按钮，表单会做出提示，显示“请填写此字段。”，效果如图2.18所示。



图2.18 required属性使用

required属性在表单的应用中最为常用，同时结合pattern属性，能发挥出强大的验证功能。

2.3.5 File对象

在HTML 5出现之前，想要通过浏览器访问本地文件只能依赖一些第三方的控件，不仅在开发上增加了难度，而且在使用上也会变得复杂。在新出的HTML 5中，根据W3C的规定，浏览器允许JavaScript获取本地文件的大小、名称、类型等多种数据。File对象的出现改善了基于浏览器应用的文件上传方式，使得文件拖放上传成为可能，从此开发者可以不需要借助Flash也能开发出完美流畅的上传体验。

File对象拥有的属性如表2.6所示。

表2.6 File对象属性

属性名称	说明
lastModifiedDate	文件最后修改时间
name	文件名称
size	文件大小，单位字节
type	文件类型

File对象拥有的方法如表2.7所示。

表2.7　File对象方法

方法名称	说明
getAsBinary	获取一个字符串，该字符串包含在原始二进制文件的数据格式
getAsDataURL	返回一个字符串，其中包含一个数据URL编码的全部内容引用的文件
getAsText	返回文件的内容作为一个字符串中，文件的数据被解释为文本使用给定的编码

下面展示如何获取File对象，代码如下：

```
<!DOCTYPE html>
<html>
<body>
      <input type="file" id="input">                  <!-- 文件选择 -->
</body>
<script type="text/javascript">
      document.getElementById("input").addEventListener("change",
 function () {  // 监听上传change事件
             console.log(this.files);                  // 打印文件信息
      }, false);
</script>
</html>
```

示例效果如图2.19所示。

选择文件 未选择文件

图2.19　File对象使用示例

单击“选择文件”按钮，选中本地图片，此时浏览器控制台输出图片相关信息，效果如图2.20所示。

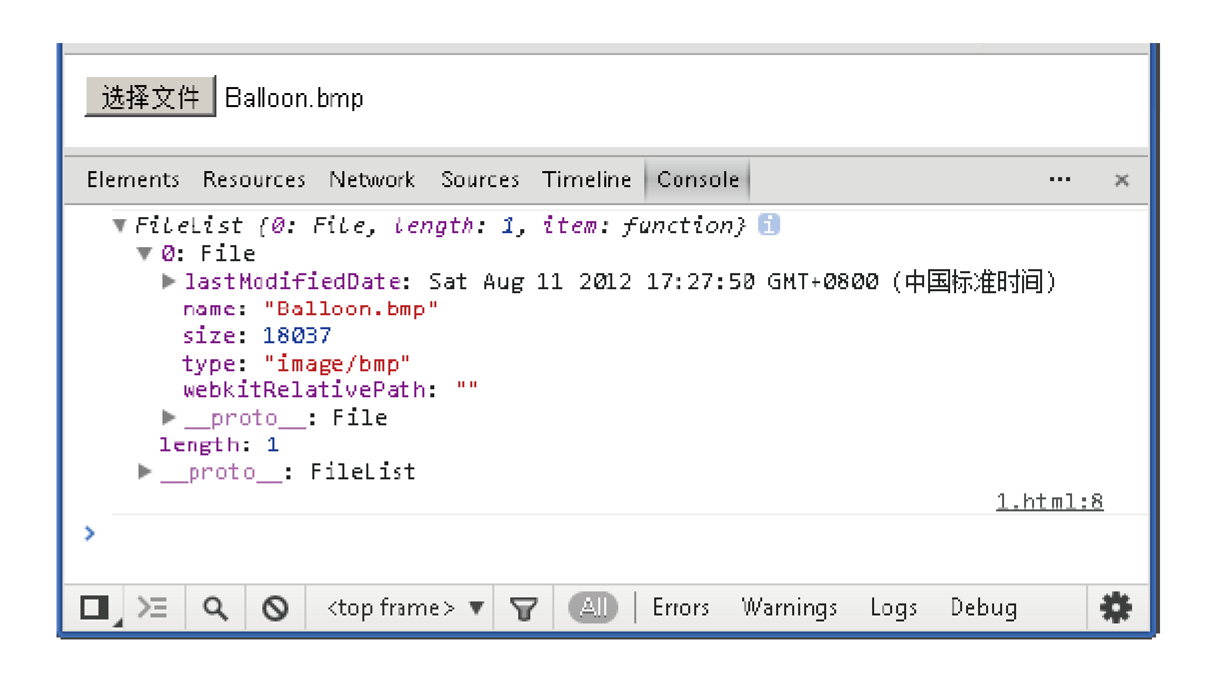


图2.20　上传文件控制台输出信息

如图2.20所示，当选中文件后触发input的change事件，回调函数获取input元素的files属性，files属性返回FileList对象，里面包含了选中文件的相关信息。如果在input上添加multifile属性，则允许用户添加多个文件，并且所有选中的文件信息均被添加到input的files属性的FileList对象中。

2.3.6 FileSystem接口

HTML 5除了提供用于获取文件信息的File对象外，还添加了FileSystem相关的应用接口。FileSystem对于不同的处理功能做了细致的分类，如用于文件读取和处理的FileReader和FileList对象、用于创建和写入的Blob和FileWriter对象、用于目录和文件系统访问的DirectoryReader和LocalFileSystem对象等，FileSystem功能的出现是浏览器在文件系统上的突破，具有里程碑的意义，虽然目前还尚未完全成熟，但足以让开发者发挥更大的想象空间。

1. FileReader对象

FileReader对象专门用于读取文件，同时可以将文件转化为各种格式信息。使用FileReader对象非常简单，FileReader对象实例一共包含4个方法，如表2.8所示。

表2.8 FileReader对象方法

方法名称	说明
readAsBinaryString	将文件读取为二进制码
readAsDataURL	将文件读取为DataURL，一段是以“data:”开头的字符串
readAsText	将文件读取为文本，第二个参数为编码类型，默认为UTF-8
abort	中断读取

下面通过示例展现FileReader对象中readAsDataURL方法的使用，代码如下：

```
<!DOCTYPE html>
<html>
<body>
      <input type="file" id="input"><br>       <!-- 文件选择控件 -->
      <img id="img"/>                          <!-- 图片展示 -->
</body>
<script type="text/javascript">
      document.getElementById("input").addEventListener("change",
  function () {// 监听选择控件change事件
             var fileReader = new FileReader();
             // 新建FileReader对象实例
             fileReader.onloadend = function(e) {
             // 监听实例loadend事件
                    document.getElementById("img").src = e.target.
result;      // 设置图片base64值
             };
             fileReader.readAsDataURL(this.files[0]);  // 读取文件内容
      }, false);
</script>
</html>
```



本节FileSystem的示例代码均在Chrome 28下测试通过。

示例打开运行效果与图2.19相同。单击“选择文件”按钮，选中本地图片，此时“选择文件”按钮下方出现对应选中图片的内容，效果如图2.21所示。



图2.21 FileReader对象readAsDataURL方法使用

示例中，当用户选中图片时，触发input元素的change事件，在回调事件中新建一个FileReader对象的实例用于读取图片文件内容，被读取的图片文件返回格式如下：

```
data:[<MIME-type>][;charset=<encoding>][;base64],<data>
```

图片被转化为DataURL数据，即Base64格式数据，该数据可以被赋予图片元素的src属性获得并显示。



Base64数据格式的说明可以参考网站http://en.wikipedia.org/wiki/Data_URI_scheme。

FileReader作为FileSystem中的一部分，通常用于文件读取，可以结合上传文件场景使用。下面将介绍FileSystem中其他的部分。

2. FileSystem接口的其他应用

在使用FileSystem对象相关应用接口时，首先要获得对沙箱文件系统的访问权限，代码如下：

```
// 获取兼容文件请求对象
window.requestFileSystem  = window.requestFileSystem || window.
webkitRequestFileSystem;
// 请求获取浏览器沙箱文件系统
window.requestFileSystem(type, size, successCallback, errorCallback)
```

window.requestFileSystem方法可以传递4个参数。

- type：包含window.TEMPORARY和window.PERSISTENT两种值，window.TEMPORARY表示存储文件数据于临时空间，共享1GB的空间。window.PERSISTENT存储于持久空间，按域名授权分配，需要进行用户许可申请。
- size：请求用于存储空间的大小，以字节计。
- successCallback：文件系统请求成功回调函数。
- errorCallback：文件系统请求失败回调函数。

FileSystem文件系统申请的空间只能作用于浏览器沙箱文件系统，同时受到域名的限制，并且不具备用户本地的任意文件夹的读写权限。当使用window.TEMPORARY申请浏览器空间不需要对用户进行授权，如创建一个5MB的临时存储空间代码如下：

```
window.requestFileSystem(window.TEMPORARY, 5*1024*1024, function(file_
system){
    console.log( ' 创建5mb临时存储空间: ' + file_system.name);
}, errorCallback);
```

如果要进行永久空间的创建，首先需要进行用户授权，浏览器询问授权时，地址栏下方会出现黄色提示确认条，如图2.22所示。

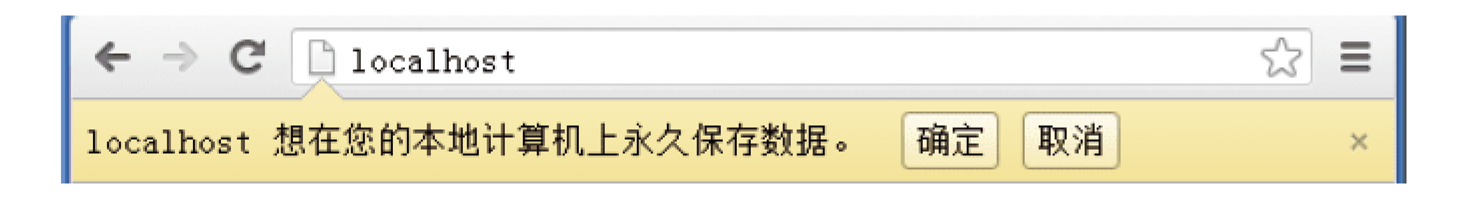


图2.22 请求浏览器永久空间提示条

请求永久空间代码如下：

```
window.webkitStorageInfo.requestQuota(window.PERSISTENT, 5*1024*1024,
 function(bytes) {
    window.requestFileSystem(window.PERSISTENT, bytes, function(file_
system){
            console.log( ' 创建5MB永久存储空间: ' + file_system.name);
    }, errorCallback);
}, function(e) {
    console.log('Error', e);
});
```

临时空间与永久空间申请时的错误回调函数相似，均返回一个FileError对象记录错误信息，不过在具体应用使用时，应该将返回的错误信息进行调整，易于用户理解，错误回调函数代码如下：

```
function errorCallback (e) {
    var msg = '';
    switch (e.code) {
    case FileError.QUOTA_EXCEEDED_ERR:      msg = '超出允许最大值'; break;
    case FileError.NOT_FOUND_ERR:           msg = '没有找到指定地址节点';
 break;
    case FileError.SECURITY_ERR:            msg = '访问拒绝'; break;
    case FileError.INVALID_MODIFICATION_ERR:      msg = '不被允许的修改请求';
break;
    case FileError.INVALID_STATE_ERR:       msg = '不被允许的执行对象接口状态';
break;
    case FileError.ENCODING_ERR:            mag = '路径不完整或者无效'; break;
    case FileError. NO_MODIFICATION_ALLOWED_ERR:  mag = '系统阻止写入文件或目
录'; break;
    case FileError. NOT_READABLE_ERR:       mag = '文件或目录没有读权限';
 break;
    case FileError. PATH_EXISTS_ERR:        mag = '文件或目录已存在相同路径';
break;
    case FileError. TYPE_MISMATCH_ERR:      mag = '错误的入口目录类型 '; break;
    default:                                msg = '未知错误'; break;
    };
    console.log('Error: ' + msg);
}
```

在使用FileSystem创建浏览器沙箱文件和目录前，笔者推荐一款用于查看当前域名沙箱文件的Chrome插件，插件通过FileSystem应用接口展示临时和永久两种类型的文件信息，并

具备清空本域名下的沙箱文件内容功能，该插件名为HTML 5 FileSystem Explorer，用户可以访问谷歌官网的安装地址https://chrome.google.com/webstore/detail/html5-filesystem-explorer/nhnjmpbdkieehidddbaeajffijockaea，插件使用截图如2.23所示。

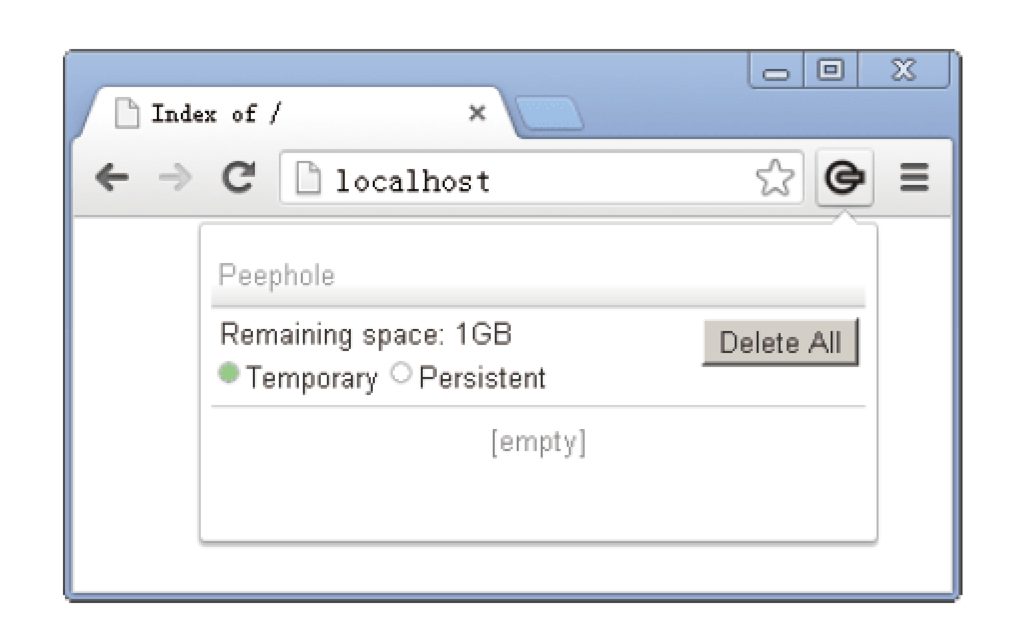


图2.23 Chrome插件HTML5 FileSystem Explorer

3. FileSystem的应用示例

下面通过创建一个临时的沙箱文件来了解FileSystem的具体使用，代码如下：

```
// 获取兼容文件请求对象
window.requestFileSystem  = window.requestFileSystem || window.
webkitRequestFileSystem;
window.requestFileSystem(window.TEMPORARY, 5*1024*1024, function(fs){
    fs.root.getFile('测试.txt', {create: true, exclusive: true},
function(file) {                               // 创建文件
          console.log( '创建文件成功: ' + file.name);        // 成功返回提示
    }, errorCallback);
}, errorCallback);
```



示例代码两处的errorCallback函数与先前相同。

代码复制到页面后，不能直接用浏览器打开该页面，需要将包含代码的页面安置于Web服务器，如IIS、Ngnix、Apache等。通过URL访问浏览器包含代码的文件地址，示例代码执行完毕以后，浏览器的localhost域名下的沙箱文件系统创建了一个空白的“测试.txt”文件，使用Chrome插件HTML5 FileSystem Explorer查看刚刚创建的文件，如图2.24所示。

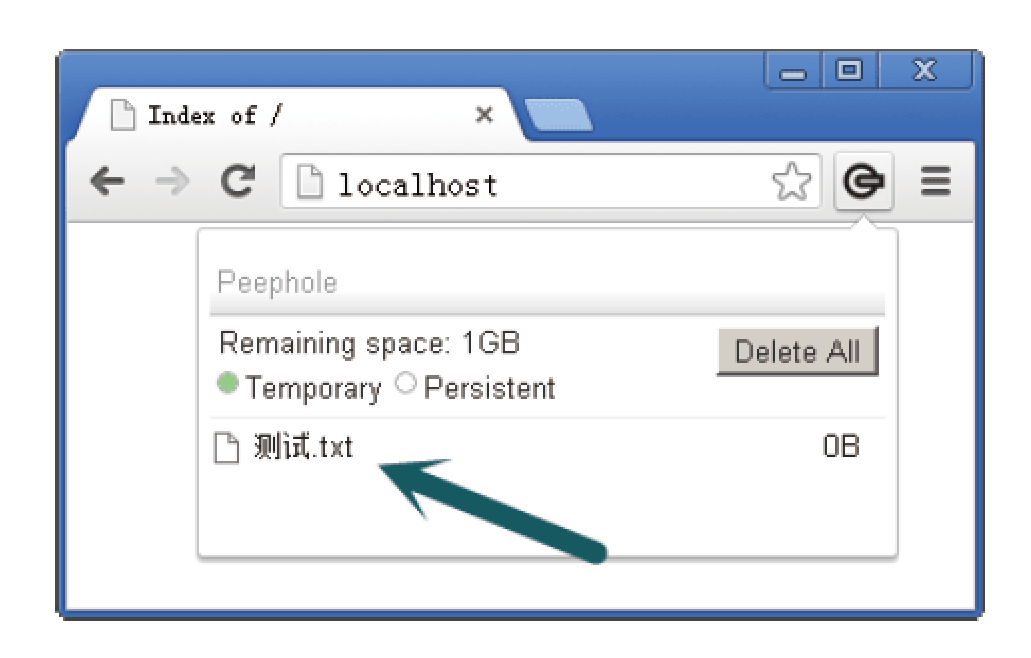


图2.24 创建“测试.txt”空文件

“测试.txt”文件创建完毕后，下面将通过FileSystem的接口向其中写入数据，代码如下：

```
window.requestFileSystem  = window.requestFileSystem || window.
webkitRequestFileSystem;
window.requestFileSystem(window.TEMPORARY, 5*1024*1024, function(fs){
    fs.root.getFile('测试.txt', {create: false}, function(file) {
                   // 获取创建的文件
          file.createWriter(function(fileWriter) {
                           // 创建写入对象
                           fileWriter.seek(fileWriter.length);
                   // 在指定位置写入
                           var blob = new Blob(['Hello World']);
                   // 初始化文本数据对象
                           fileWriter.write(blob);
                           // 写入数据
          });
    });
});
```

执行完写入脚本以后，使用Chrome插件HTML 5 FileSystem Explorer查看刚刚写入的“测试.txt”文件，效果如图2.25所示。

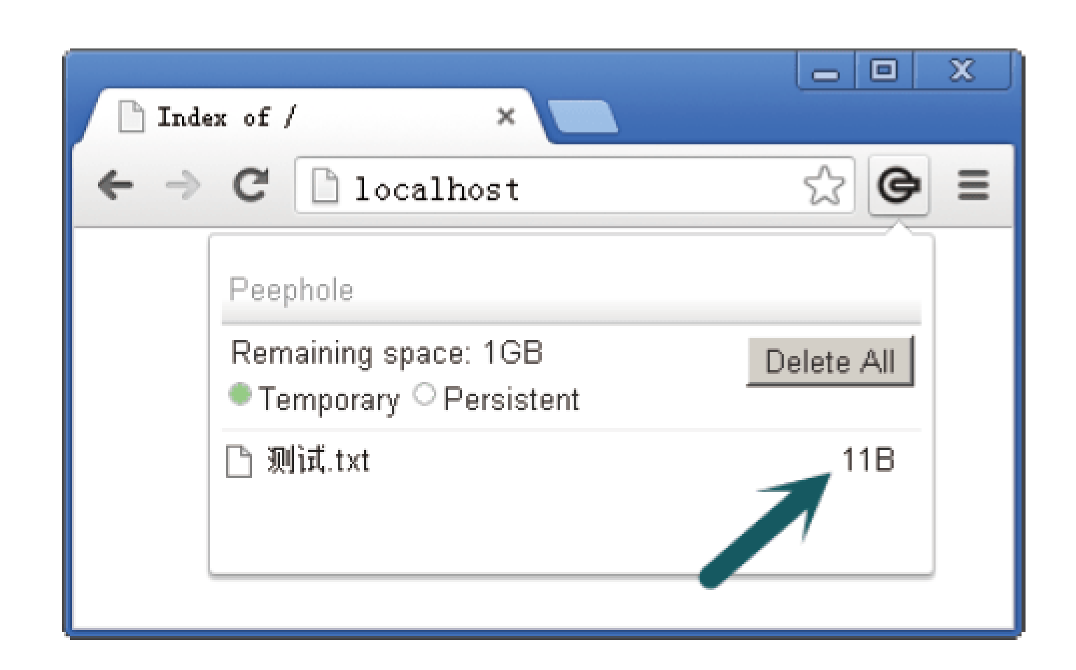


图2.25 向“测试.txt”文件写入数据

通过插件看到，“测试.txt”文件被写入了11字节的数据，即Hello World共11字节。同时可以在浏览器地址栏内输入“filesystem:http://localhost/temporary/测试.txt”进行访问，效果如图2.26所示。

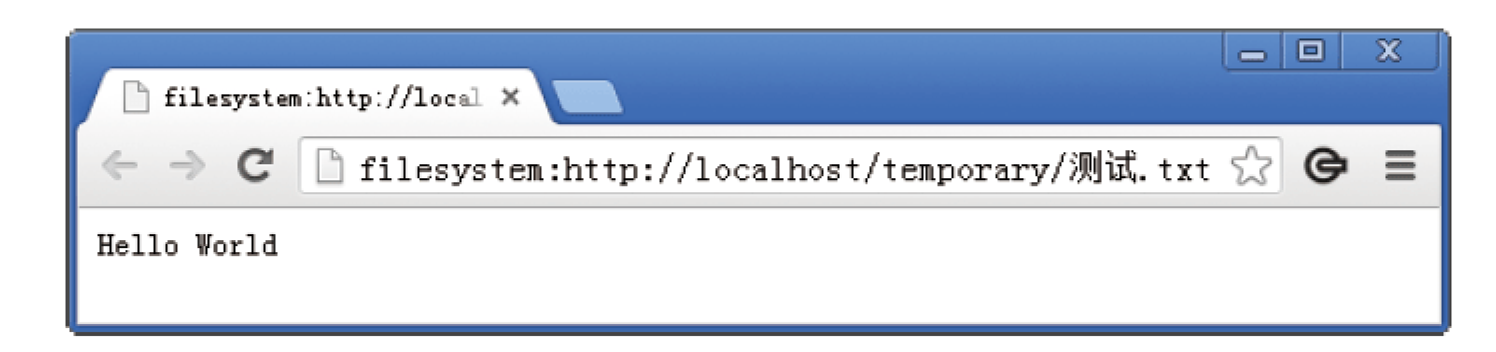


图2.26 通过浏览器访问文件系统内容

最后使用FileSystem对象的接口将“测试.txt”文件文件删除，代码如下：

```
window.requestFileSystem  = window.requestFileSystem || window.
webkitRequestFileSystem;
```

```
window.requestFileSystem(window.TEMPORARY, 5*1024*1024, function(fs){
    fs.root.getFile('测试.txt', {create: false}, function(file) {
                // 获取文件
        file.remove(function() {                        // 删除文件
                console.log('文件删除成功');             // 删除成功
        });
    });
});
```

执行脚本后，文件删除成功，此时使用Chrome插件HTML 5 FileSystem Explorer查看效果如图2.23所示，沙箱文件系统被清空。

刚刚从文件的新建、修改、删除三个方面介绍了FileSystem接口的应用，实际上只是FileSystem接口应用的很小一部分，更多的使用方法和场景可以参考本书的示例部分。如此强大的FileSystem接口，让Web开发人员拥有了更大的施展和发挥舞台，同时HTML 5与文件相关的应用操作还可以解决实际开发中的很多问题，下面例举了部分应用场景：

- 实现断点续传。读取本地文件保存至浏览器的文件系统沙箱内，并逐块上传，如果系统异常关闭，再次打开时也能从浏览器文件系统中获取，并继续上传。
- 存放网络游戏中的图片和相关资源。将游戏中新的关卡和功能在运行后台时自动加载至浏览器本地文件系统，当游戏调用时可以直接读取文件，大大减少了服务器的消耗，同时达到比网络下载离线快得多的速度。
- 离线的图片编辑功能。开发的图片网页应用，可以将编辑完的图片直接存入浏览器文件系统，待下次使用时可以直接应用。
- 离线的视频播放器。可以在没有下载完整个视频时进行部分播放，同时还能选择在不同的时间点之间进行选择播放，截取视频的一部分。
- 离线的邮箱系统。保存邮件中的附件内容，当浏览器处于断网状态，可以将附件和邮件内容进行保存，待联网时重新进行发送。

以上畅想的内容远不止未来HTML 5所带来的全部面貌，以往无法实现的功能将发生改变，未来的浏览器一切皆有可能。

2.3.7 jQuery html5Validate HTML 5表单验证插件

之前的章节已经对HTML 5新增的表单元素和属性做过介绍，极大地增强了开发体验，同时也给传统的表单验证插件开发提供了方向。未来的表单设计应该遵循W3C的表单验证规范，本节将介绍的jQuery html5Validate正式基于这种设计开发的新一代表单验证插件。

首先该款jQuery html5Validate插件是基于jQuery并由国人设计的表单验证插件，设计秉承了HTML 5表单的设计风格，如HTML 5中要表示一个文本输入框为必填时，代码如下：

```
<input type="number" required>
```

插件的HTML结构与原生语法相同，使用时只需要引入对应的脚本和添加插件初始化脚本，代码如下：

```
<!DOCTYPE html>
<html>
```

```
    <head>
        <script src="jquery.min.js"></script>    // jQuery脚本
        <script src="jquery-html5Validate.js"></script>
                // 插件脚本
    </head>
<body>
    <form>
        <input type="number" required>          // 不为空输入框
    </form>
    <script>
        $("form").html5Validate();               // 插件初始化表单
    </script>
</body>
</html>
```

html5Validate插件支持目前所有HTML 5的input类型验证，如类型为email、number、tel、url，同时还支持HTML 5的验证属性，如step、min、max、required、patern、multiple等，另外还扩展了部分类型，如zipcode（表示输入类型为邮编）、zhcn（表示输入类型为中文）等。

html5Validate在初始化表单时，还可以选择添加配置参数，默认值为true，如表2.9所示。

表2.9 html5Validate初始化配置参数

参数名称	说明
novalidate	布尔型。是否取消现代浏览器的内置验证
submitEnabled	布尔型。表单中禁用的提交按钮是否使之可用
labelDrive	布尔型。是否优先使用label标签中的文字作为提示关键词
validate	包含返回值的函数，插件自带验证以外的其他脚本或者验证

参数使用代码如下：

```
$("form").html5Validate(function() {
    console.log("验证通过");
},{
    novalidate : true,
    submitEnabled : true,
    labelDrive : true
    validate : function() {
          return true;
    }
});
```

jQuery html5Validate在设计上相当巧妙，避开了传统表单验证插件设计时过多的脚本参数配置，一切遵循W3C的规范，是国内表单插件设计中的佼佼者。



jQuery html5Validate插件文档地址为http://www.zhangxinxu.com/wordpress/?p=2857，插件示例可以参考http://www.zhangxinxu.com/study/201212/jquery-html5validate-plugin-test.html。



2.4 图形绘制

HTML语言经历20多年的发展，现今已经成长为最广泛的编程语言，在互联网上随处可见，虽然HTML具备很多优点，但始终没有完全解决图形绘制问题，并且没有提供一个成熟的解决方案，这也是Flash在互联网领域仍存在的根源。在HTML 5出现之前，HTML已经存在一些技术可以绘制图形，常用的基于XML的技术绘制，如VML和SVG，都能够良好地支持矢量图形在Web页面上的显示，同时提供一些事件和动态机制。后来，HTML 5 Canvas出现后，从某方面弥补了过去的不足，基于画布的绘制，让开发者更容易操作页面上的像素级内容。不过由于历史原因，市面上各种浏览器兼容情况不同，对于要求兼容众多浏览器和终端设备的应用，可以通过浏览器类型判断，如VML在IE中使用，其他不支持Canvas的浏览器采用SVG，笔者推荐使用第三方开源类库解决图形绘制问题，如Raphael图形类库和本节将要介绍的Paper.js矢量图形类库。

2.4.1 Canvas是什么

Canvas是HTML 5新增的元素特性，允许脚本语言动态渲染绘制图形，如可以用Canvas画图、合成图像或者制作动画。

Canvas最早在苹果公司的Mac OS X Dashboard上被引入，后来被内置于Safari浏览器内。之后在Firefox、Opera、Chrome的推动下，W3C将Canvas纳为HTML 5的一部分。IE在Canvas的支持上仍然比其他浏览器慢了一步，直到IE 9才开始支持Canvas元素。



要了解苹果公司的Mac OS X Dashboard可前往网站http://www.apple.com/macosx/features/dashboard/。

Canvas由绘制区域HTML代码的属性决定宽高，同时JavaScript可以访问Canvas元素区域，通过Canvas提供的一套完成绘图应用程序接口生成图形。Canvas与VML和SVG最大的区别在于，Canvas有一套完全基于JavaScript脚本语言绘制的应用程序接口，而VML和SVG使用XML文档描述绘制图形。

2.4.2 什么情况下用Canvas

HTML 5出现后，图形绘制使用方面开发者一直都拿SVG与Canvas进行比较，下面将两项技术做一个对比，如表2.10所示。

表2.10 SVG与Canvas比较

SVG	Canvas
与分辨率无关，放大或缩小图形质量不会下降	可以控制画布上的每个像素，绘制出的是位图
多个元素节点组成	单个Canvas元素
图形允许CSS和脚本修改	图形完全由脚本修改

（续表）

SVG	Canvas
对动画支持较容易，可以通过SVG语法描述	制作动画需要不断重复绘制画面
导出图片需要借助其他技术实现	可以方便地直接在导出jpg或png格式图片

技术的选择对于开发后期的维护起到了至关重要的作用，通常情况下，当应用需要处理点阵图时，如切割图片、去除红眼，那么请选择使用Canvas。如果想开发一款对速度要求较高的网页游戏，也可以选择使用Canvas，如比较著名的游戏Cut The Rope（中文名割绳子）使用Canvas开发了一个测试版本，地址为http://www.cuttherope.ie/。但是遇到一些数据可视化的图表时，要求图形能在不同分辨率下正常显示，应使用SVG，这时候使用Canvas开发就不是一个明智的选择。

2.4.3 检测浏览器对Canvas的支持情况

检测浏览器是否支持Canvas操作通常使用特征检测法，即判断检测对象上是否含有特定的属性或者方法。在Canvas对象上都会包含一个名为getContext的方法，如果浏览器支持Canvas，就代表该方法存在，检测代码如下：

```
function is_supports_canvas(){
    return !!document.createElement('canvas').getContext;
}
```

不过由于历史的原因，Canvas的检测并不会如此简单。市面上有些浏览器实现了部分Canvas应用接口，但没有加入与Canvas Text相关的功能，那么使用上述方法检测后仍然无法完整使用Canvas。下面要做的是对刚才的方法进行强化，增强后的代码如下：

```
function is_supports_canvas(){
    var cvs = document.createElement('canvas');
    if(!cvs.getContext )      return false;
    // 首选判断getContext方法是否存在
    return typeof cvs.getContext('2d').fillText  == 'function';
// 判断是否含有fillText方法
}
```

通常检测浏览器是否支持指定的HTML 5功能，按先后顺序分为4个步骤，前面曾经介绍过，这里再复习一遍：

（1）检查浏览器全局对象是否支持指定的对象属性。
（2）创建元素，检查元素是否存在指定的属性。
（3）创建元素，检查元素是否存在指定的方法，同时检查方法返回值。
（4）创建元素，给元素设定HTML 5的特有属性值，检测被赋值的属性是否保存了设定值。

2.4.4 在页面中加入Canvas

Canvas在页面上的使用从元素定义开始，代码如下：

```
<canvas width="150" height="150"></canvas>
```

Canvas元素拥有width和height属性，两个属性均可选，并且可以用DOM属性或者CSS来设置，如果不指定宽高，则会默认为宽300像素和高150像素，通过CSS设置Canvas的宽高有的时候会出现渲染变形，建议尝试采用设置Canvas的width和height属性。

很多老一代的浏览器不支持Canvas，这时候需要对不支持Canvas的浏览器做出提示。提示的设置非常简单，只需要在Canvas的元素内插入提示文本内容，不支持Canvas的浏览器会将其识别为未知标签兼容渲染成为文字信息，而支持Canvas的浏览器会做出正确的渲染，代码如下：

```
<canvas width="150" height="150">您的浏览器不支持canvas，请升级后再使用！
</canvas>
```

当Canvas被添加到页面上后，初始化渲染是空白的，想要在上面通过脚本进行图形绘制首选需要获得渲染的上下文，通过Canvas元素对象的getContext方法获得，该方法在上一节浏览器Canvas检测的时候已经学习过如何使用。

下面是使用Canvas绘制一个矩形的示例，代码如下：

```
<html>
<head>
      <script>
      window.onload = function() {                          // 资源加载结束后触发
             var canvas = document.querySelector("canvas");  // 获取canvas元素
             if (canvas.getContext) {
                    var ctx = canvas.getContext("2d");       // 获取渲染上下文
                    ctx.fillStyle = "rgb(200,50,0)";         // 填充颜色
                    ctx.fillRect (50, 50, 50, 50);      // 绘制矩形
             }
      }
      </script>
</head>
<body>
      <canvas width="150" height="150">您的浏览器不支持canvas，请升级后再使用！</canvas>
 </body>
</html>
```

示例的运行效果如图2.27所示。

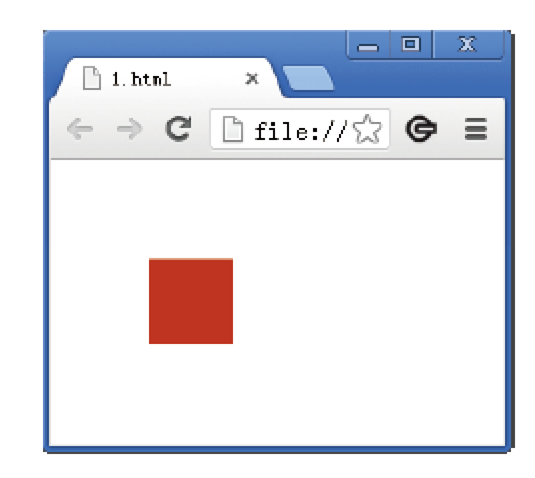


图2.27 使用Canvas绘制一个矩形

2.4.5 SVG是什么

上一节一直将SVG拿来与Canvas做比较，大致可以了解到SVG可以用来处理哪些问题。SVG是Scalable Vector Graphics的缩写，中文意思是可缩放矢量图形，是基于XML（可扩展标记语言）用来描述二维矢量图形的一种图形格式。

SVG诞生于2000年的8月，由W3C（国际互联网标准组织）制定，由于采用文本格式的描述性语言来渲染图片，因此产生的图片和图像分辨率无关，即使缩放图形质量也不会下降。SVG有如下优点：

- 基于XML，继承了XML的扩平台和可扩展的特性。
- 采用文本描述图形对象，利于搜索引擎通过文本内容搜索图片信息。
- 良好的交互和动态特性，可以在其中嵌入动画，通过脚本收缩、旋转调整图形。
- 对DOM支持完整，可以通过脚本获取元素，监听元素事件。
- 体积小下载快，与GIF和JPG格式的图片相比具有较小的体积，在互联网上传输有明显优势。

2.4.6 SVG的使用

由于SVG不是本书的重点，下面通过一个简单的示例介绍SVG使用的初步功能，代码如下：

```
01  <?xml version="1.0" standalone="no"?>
02  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/
Graphics/SVG/1.1/DTD/svg11.dtd">
03  <svg width="100%" height="100%" version="1.1" xmlns="http://www.
w3.org/2000/svg">
04        <circle cx="100" cy="100" r="50" stroke="black" stroke-width="2"
fill="blue"/>
05  </svg>
```

以svg为后缀保存文件，使用Chrome浏览器打开，效果如图2.28所示。

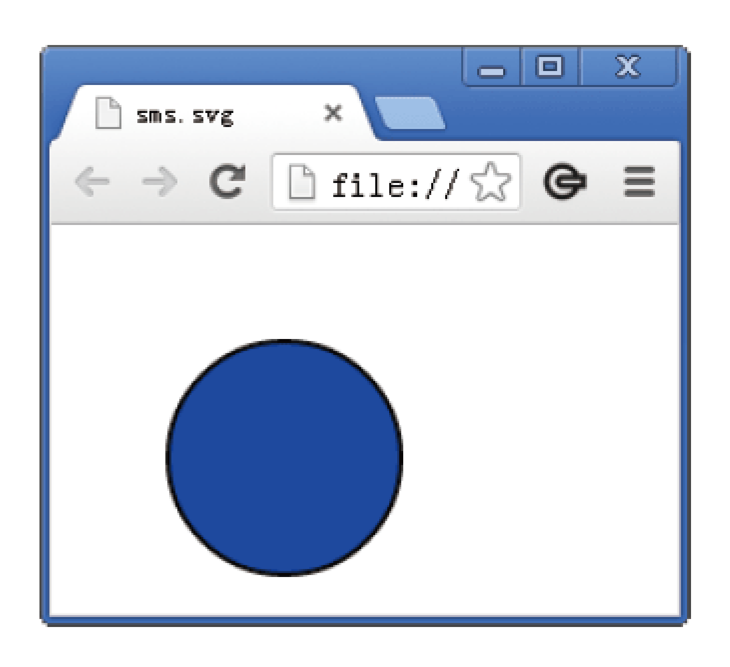


图2.28 使用SVG画圆

示例代码虽然简单，但是包含了很多比较陌生的属性和节点信息。

第01行，里面定义了XML文件的声明，这里有一个很关键的属性standalone，表示该SVG文件是否引用外部文件，示例中standalone属性被赋值为no意味着该文件会引用一个外部

文件，地址即“http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd”。

上面提到的DTD文件“http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd”主要用于SVG规范，里面包含了所有SVG允许使用的元素。

第03行是整个绘图SVG文件的根元素svg，类似于HTML文件中的html根元素。width和height属性定义SVG的宽高，version属性定义使用SVG的版本，xmlns属性定义SVG的命名空间。

第04行的circle元素用来创建一个圆。cx和cy属性定义圆中心的x和y轴坐标，默认设置均为0。r属性用来定义圆的半径。stroke和stroke-width属性用来设置显示图形的轮廓，stroke这里设置为黑色边框，stroke-width设置为2像素的边框。fill属性用来设置图形内填充的颜色。



所有标签的规则必须严格遵循W3C规范，开启标签必须有对应的闭合标签。

2.4.7 WebGL是什么

WebGL规范由Khronos Group协会在2011年3月的美国洛杉矶举办的游戏开发大会上发布，该标准允许把JavaScript和OpenGL ES 2.0相结合为HTML 5 Canvas提供硬件3D加速渲染，使开发人员可以借助系统显卡在浏览器里展现3D场景和模型，使用者可以在不安装插件的情况下体验3D图形技术。



Khronos Group成立于2000年1月，由包括3Dlabs、ATI、Discreet、Evans & Sutherland、Intel、Nvidia、SGI和Sun Microsystems在内的多家国际知名多媒体行业领导者创立，致力于发展开放标准的应用程序接口，以实现在多种平台和终端设备上进行富媒体创作、加速和回放。

目前WebGL标准已经在Firefox、Safari、Chrome浏览器上得到了支持，微软将在IE 11中加入WebGL功能，相信不久的将来开发者可以通过WebGL展现各种3D模型和场景，推出更多基于3D的网站和游戏。

2.4.8 WebGL的使用

WebGL对于Web开发者来说，提供前所未有的想象空间，打开了一个面向3D技术的新领域，从此不需要借助Flash或Silverlight等浏览器插件也能制作出丰富的视觉交互体验。目前已经涌现出很多使用WebGL技术开发的非常厉害的作品。

谷歌推出了WebGL版的Google Map，用户可以前往地址https://maps.google.com/查看。进入网站后，网页会判断浏览器支持WebGL的情况，当确认用户的浏览器支持WebGL时，页面左侧会出现提示，如图2.29所示。

使用了WebGL技术后，地图效果得到极大的提升，增加了3D图形显示，地图拖动切换更加平滑，同时还带有45°的视角旋转功能。

另外谷歌还在Chrome实验室里提供了很多WebGL的实验产品，让人叹为观止，欣赏作品请前往站点http://www.chromeexperiments.com/webgl。其中有一款关于Google Maps的游戏，相信一定会让用户眼睛为之一亮，游戏地址为http://www.playmapscube.com/，游戏截图如图2.30所示。

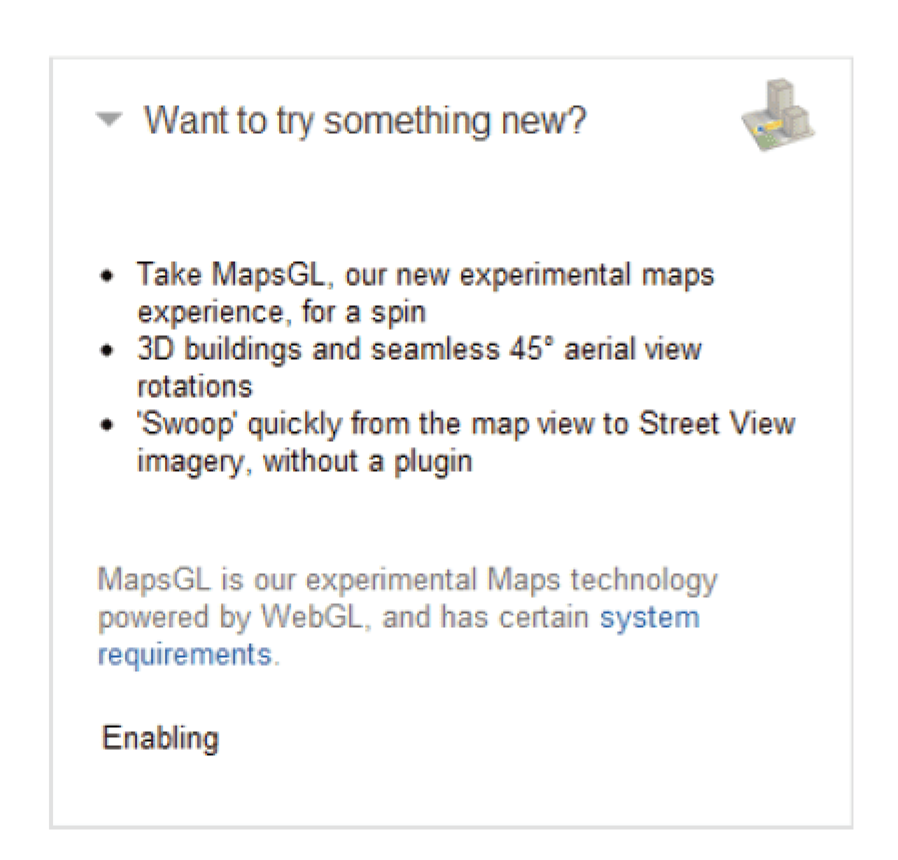


图2.29 谷歌地图开启WebGL功能提示

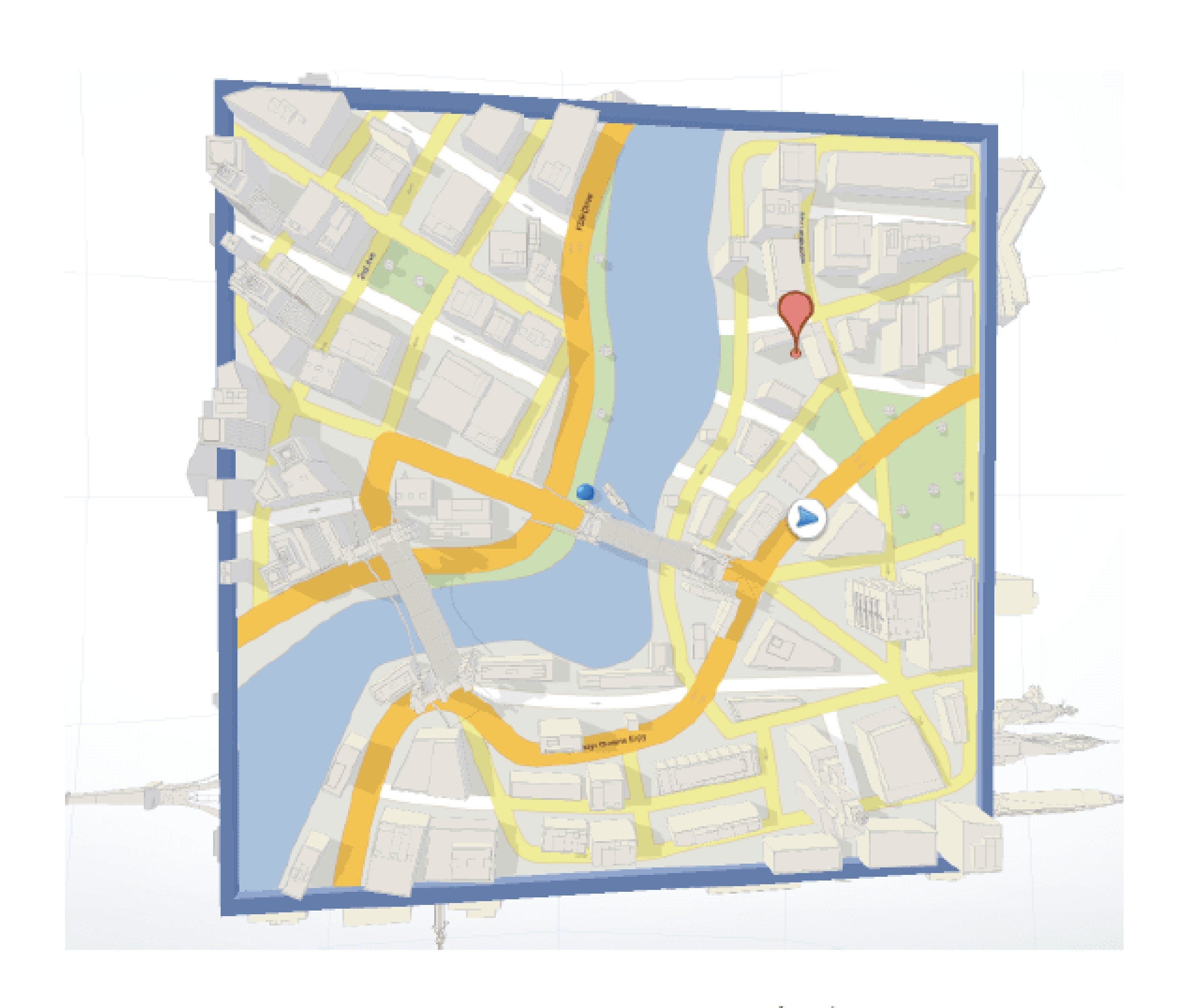


图2.30 Google Maps地图小游戏

2.4.9 Paper.js图形库

Paper.js是一个开源的矢量图形脚本框架，基于HTML 5 Canvas开发。提供了清晰的场景图和文档对象模型，还有许多强大的功能用来创建和使用矢量图形和贝塞尔曲线，接口设计精巧、规范并且干净。

Paper.js的中文教程目前还很少，所以使用者需要前往官网http://paperjs.org的英文版教程学习使用，这对一般的使用者是一个挑战，不过作为一名合格的前端工作者，习惯阅读外文站点也是工作的一部分。下面通过一个示例了解使用Paper.js，开发使用Canvas的图形绘制应用，代码如下：

```
<!DOCTYPE html>
<html>
<head>
     <script type="text/javascript" src="paper.js"></script>
     <!-- 引入paper.js -->
     <script type="text/paperscript" canvas="canvas">
     <!-- 执行脚本块 -->
             function onMouseDrag(event) {
         // 鼠标拖动事件
                  var path = new Path.Circle({
         // 新建一个圆形路径实例
                        center: event.downPoint,
         // 设置中心点坐标
                        radius: (event.downPoint - event.point).length,  // 设置圆形半径
                        fillColor: 'white',
         // 设置填充颜色
                        strokeColor: 'black'
         // 设置边框颜色
```

```
                            });
                     };
        </script>
</head>
<body>
        <canvas id="canvas" resize></canvas>            // 画布元素
</body>
</html>
```

示例使用时，单击页面某个区域，不松开鼠标进行拖动，鼠标会以单击点为圆心，拖动距离为半径画圆，松开鼠标后圆形绘制完毕，图2.31为笔者测试后绘制的效果图。

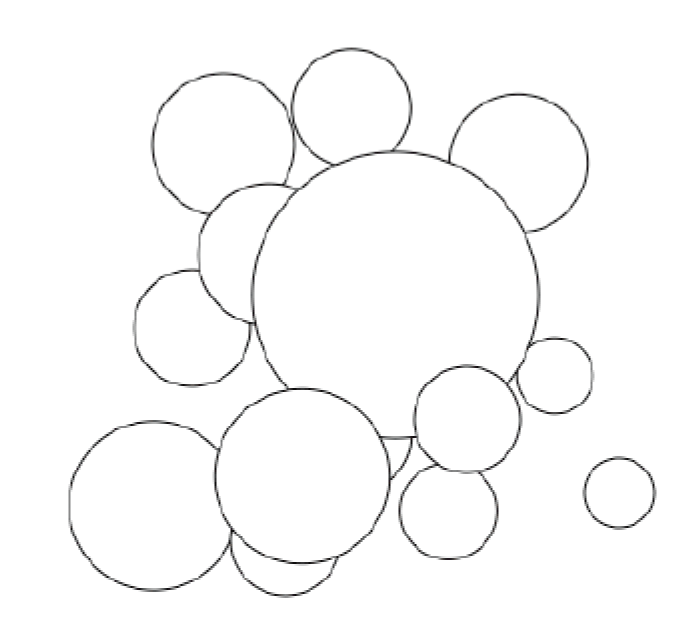


图2.31 使用Paper.js示例绘制效果题



Paper.js项目同时还被发布在GitHub上，了解开发者最新发布更新动态可以前往GitHub，项目地址为https://github.com/paperjs/paper.js。



2.5 音频视频

音频视频作为HTML 5的最大亮点之一，解决了常年以来使用浏览器观看视频和音频都不得不安装Flash的问题。随着HTML 5的发展，各大浏览器厂商对音频视频的支持，加上苹果的iOS系统禁止使用Flash，使得HTML 5的音频视频在这几年内得到了飞速的发展，各大主流视频网站纷纷推出了完全使用HTML 5打造的视频编辑器。使用HTML 5的音频和视频非常简单，只需要用到两个标签audio和video，本节将介绍这两个标签的使用和注意事项。

2.5.1 音频和视频编码解码器

目前，所有主流的浏览器都支持audio和video元素，包括微软的IE 9和之后的版本，但是不同的浏览器支持的编码解码器各不相同，幸运的是HTML 5在设计之初考虑到了这一点，并且提供了非常灵活的方式来使用音频和视频功能。

通过一个示例代码，展示在各种新老版本浏览器上如何使用视频的功能，代码如下：

```
<video>
    <source src="video.mp4" type="video/mp4">
    <source src="video.webm" type="video/webm ">
    <source src="video.ogv" type="video/ogv ">
    <object type="application/x-shockwave-Flash" data="Flash.swf">
          <param name="movie" value="Flash.swf" />
          <param name="Flashvars" value="file=video.mp4" />
          您的浏览器不支持播放视频功能。
    </objet>
</video>
```

MP4格式在日常的生活中已经随处可见，WebM和Ogv大家应该还相对陌生。WebM是一个开放、免费的媒体文件格式，最早由谷歌提出，该格式容器中包括了VP8和Ogg Vorbis音轨。WebM格式效率非常高，可以在平板电脑和其他一些手持设备上流畅的使用。Ogv即带有Thedora视频编码和Vorbis音频编码的Ogg文件，该格式文件带有不确定的版权问题，可能在未来的浏览器中被慢慢淘汰。

各种格式的优缺点不一，如WebM格式，依赖于Google和YouTube的推广，并且在硬件上有良好的支持，但是由于涉及到MPEG LA的专利案件，并且在iOS设备到得不到支持。虽然传统视频和音频编码技术经历多年的发展，并且相当稳定，但对于浏览器，原生支持视频和音频还非常年轻，仍然会遇到重重阻碍，不过规范和标准日益完善，如果读者及早的在视频和音频上做好技术准备，在未来会得到加倍的回报。

2.5.2 使用脚本控制播放

HTML 5除了提供audio和video元素播放音频和视频资源外，同时还配套提供了一系列的方法、属性和事件，这些方法、属性和事件允许使用JavaScript操作audio和video对象。

audio和video对象均提供了一些相类似的方法，如表2.11所示。

表2.11 audio和video对象方法

方法名	说明
load	重新加载音频或视频内容
play	播放音频或视频
pause	暂停音频或视频
addTextTrack	向音频或视频添加字幕
canPlayType	检测浏览器是否支持音频或视频格式

下面通过一个简单的视频播放示例介绍部分API的使用，代码如下：

```
<!DOCTYPE HTML>
<html>
      <body>
             <video src="video.webm" width="480" height="320" 
controls></video>       // 视频播放元素
             <a href=";" class="play">播放</a>         // 播放按钮
             <a href=";" class="pause">暂停</a>        // 暂停按钮
      </body>
```

```
        <script>
            var video = document.querySelector('video');
            // 获取视频元素
            document.querySelector('a.play'),.addEventListener('click',
function(e){    // 监听播放按钮单击事件
                e.preventDefault();         // 阻止元素默认事件
                video.play();               // 播放视频
            },false);
            document.querySelector('a.pause').addEventListener('click',
function(e){    // 监听暂停按钮单击事件
                e.preventDefault();         // 阻止元素默认事件
                video.pause ();             // 暂停视频
            },false);
        </script>
    </html>
```

该示例是一个最基本的使用JavaScript操作视频元素的例子，其中用到了两个关键方法play和pause。audio和video元素除了新增许多新的方法外，同时还增加了诸多的属性，如表2.12所示。

表2.12　audio和video元素属性

元素名	说明
autoplay	表示视频或音频加载完毕后自动播放
controls	表示显示元素浏览器默认控件条
height	视频或音频元素的高
width	视频或音频元素的宽
loop	表示视频或音频是否循环播放
preload	表示视频或音频在页面加载时自动进行加载，并预备播放
src	表示视频或音频的地址URL

在实际开发中，使用JavaScript操作视频音频元素往往会遇到很多浏览器的差异，这些问题在本章对应的示例章节会给出相应的解决方案，同时读者可以使用目前比较成熟的第三方视频音频类库解决兼容问题，如目前比较流行的基于HTML 5的类库video.js，官网地址为http://www.videojs.com/。

2.5.3　audio元素和video元素的浏览器支持情况

使用audio和video元素的第一步是要了解浏览器的支持情况，就目前而言，Safari和Chrome的支持情况最好，Firefox和Opera次之，IE表现最差。表2.13给出了目前浏览器的支持状况。

表2.13　主流浏览器对audio和video支持情况

浏览器	Audio元素	Video元素
Chrome 15+	支持	支持
Safari 5.1+	支持	支持
Firefox 8+	支持	支持
Opera 11.1+	支持	支持
IE 9+	支持	支持

目前主流的三种视频格式有MP4、WebM和Ogg，表2.14列出了主流浏览器的支持情况。

表2.14 主流视频格式的浏览器支持情况

浏览器	MP4	WebM	Ogg
IE 9+	支持	不支持	不支持
Chrome 6+	支持	支持	支持
Firefox 3.6+	不支持	支持	支持
Safari 5+	支持	不支持	不支持
Opera 10.6+	不支持	支持	支持

主流的音频格式有三种：MP3、Wav、Ogg，表2.15列出了主流浏览器的支持情况。

表2.15 主流音频格式的浏览器支持情况

浏览器	MP3	Wav	Ogg
IE 9+	支持	不支持	不支持
Chrome 6+	支持	支持	支持
Firefox 3.6+	不支持	支持	支持
Safari 5+	支持	支持	不支持
Opera 10+	不支持	支持	支持

另外，移动支持设备在未来是一个关键的发力点，目前的情况也相当得复杂，读者有兴趣可以查看文章http://access.ecs.soton.ac.uk/blog/synotemobile/2012/09/03/html-5-video-and-support-in-mobile-browsers/，了解关于Android和iOS系统下的浏览器在音频和视频元素上的支持情况。

2.5.4 音视频的实时通信

音视频的实时通信即HTML 5的WebRTC技术，是Web Real-Time Communication的缩写，该技术主要用于支持浏览器进行实时的语音对话和视频通信。

在2011年之前，浏览器实现语音对话和视频通信技术需要通过安装插件或者客户端等一些技术实现，不论对于用户还是开发人员都是一个繁琐和复杂的过程，并且还受到各种专利的影响。谷歌公司在2010年收购了Global IP Solutions公司从而获得了WebRTC技术，在2011年，按照BSD协议把该技术开源，同年W3C将WebRTC技术纳入HTML 5成为标准的一部分。最新Android系统上的Chrome版本也加入了WebRTC技术。

WebRTC技术可以让Web开发者轻松的基于浏览器就能开发出丰富的实时媒体应用，帮助网页应用开发语音通话、视频聊天、P2P文件分享等功能，而不需要安装任何插件，同时开发者也不需要关心多媒体的数字信号处理过程，只需要使用JavaScript即可实现，图2.32为WebRTC的技术架构图。

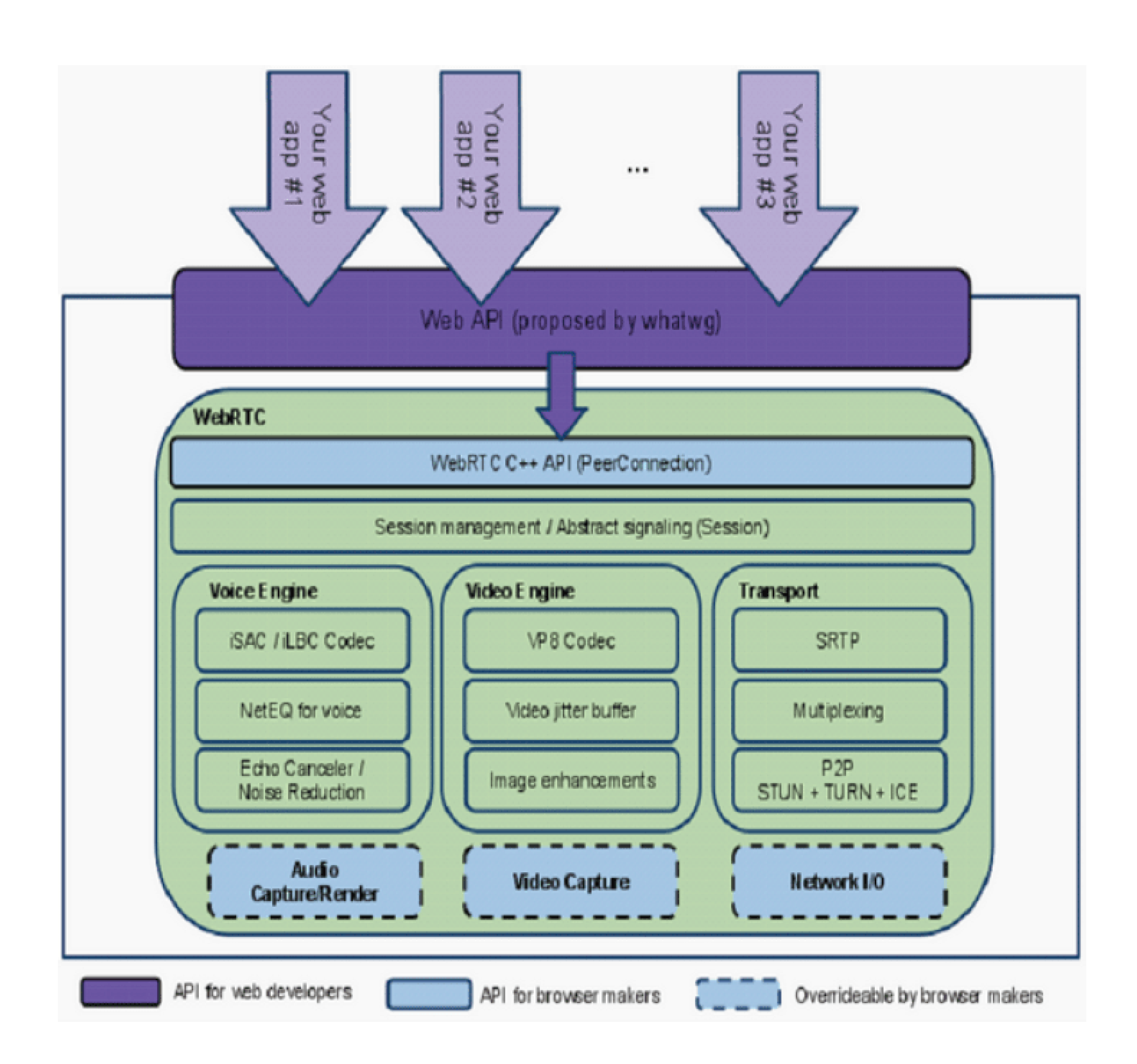


图2.32 WebRTC的技术架构题

WebRTC技术由三部分组成。

- MediaStream：本地的音频视频流或来自远端浏览器的音频视频流。
- PeerConnection：执行音频视频调用，支持加密和带宽控制。
- DataChannel：采用点对点传输，传输常规数据。

下面通过一个示例演示如何使用浏览器WebRTC，代码如下：

```
<!DOCTYPE html>
<html>
<body>
  <video autoplay></video>                    <!-- 视频播放元素 -->
  <script>
      try {                    // 使用WebKit核心下的getUserMedia方法
          navigator.webkitGetUserMedia({audio: true, video: true},
 successCallback, errorCallback);
      } catch (e) {
          navigator.webkitGetUserMedia("video,audio",
successCallback, errorCallback);
      }
      function successCallback(stream) {      // 成功回调并设置video元素
          document.querySelector('video').src = window.webkitURL.
createObjectURL(stream);
      }
      function errorCallback(error) {         // 失败回调返回错误信息
          console.log('发生错误，编号：' + error.code);
```

```
        }
        </script>
</body>
</html>
```

将上述代码保存至后缀为html的文件，并放置于Web服务器，如IIS、Apache、Nginx等。使用最新Chrome浏览器打开页面地址，浏览器会提示是否启用摄像头和麦克风，如图2.33所示。



图2.33 浏览器提示是否启用摄像头和麦克风

单击浏览器提示条中的“允许”按钮，此时浏览器内出现一个宽640像素、高480像素的视频窗口，显示内容为用户摄像头拍摄视频。

随着WebRTC的发展和各大技术巨头的支持，虽然标准尚未完全成熟，但足以给开发者带来前所未有的惊喜，Web开发人员可以完全基于浏览器开发音频视频实时在线应用。目前，已经出现了一批颇具实力的类库，如webRTC.io和WebRTC-Experiment等。



BSD协议是Berkeley Software Distribution的缩写，中文意思为伯克利软件发行版，是一整套软件发行版的统称，是自由软件中使用最广泛的许可证之一。BSD最初所有者是加州大学董事会。该协议可以自由的使用并且修改源代码，也可以将修改后的代码作为开源或者专利软件再发布。



2.6 地理位置

地理位置是HTML 5非常重要且诱人的特性之一，尤其在当今移动互联网时代更显得价值连城。开发者只需要简单的几行代码就可以轻松获取用户的地理位置信息，借助这些信息可以开发基于位置信息的高级应用，将虚拟世界和现实世界整合在一起，以一种难以捉摸、变化莫测的方式出现在大众的眼前。本节将向读者介绍HTML 5中这一项伟大的技术。

2.6.1 纬度和经度坐标

纬度和经度是一种利用三度空间的球面来定义地球上空间的球面坐标系统，能够标示地球上的任何一个位置。

谈到经纬度，可以追溯到公元前344年，亚历山大渡海南侵，随军的地理学家尼尔库斯沿途搜索材料，准备绘制一幅世界地图。尼尔库斯发现沿着亚历山大东征的路线，由西向东，无论季节变换与日照长短都很相仿。于是第一次在地球上划出了一条纬线，这条线从直布罗陀海峡起，沿着托鲁斯和喜马拉雅山脉一直到太平洋。

经线又称为子午线，定义为地球表面连接南北两极的半圆弧。任何两根经线的长度相同，相交于南北两极，每根经线都有相对应的值，称为经度。纬线定义为地球表面某个点随着地球自转所形成的轨迹，任何一根纬线都是圆形而且两两平行。

经1884年国际会议协商，决定以通过英国伦敦格林尼治天文台（原址）的经线为起始线，称为本初子午线。以本初子午线为起点，向东为东经度（E），向西为西经度（W）。经度共360°，本初子午线为0°经线，东西经度各为180°，东、西经180°经线为同一条经线，统称180°经线。

纬度以赤道为起点，赤道以北为北纬度（N），赤道以南为南纬度（S）。赤道是0°纬度，北纬度的最大值是90°，即北极点；南纬度的最大值为90°，即南极点。

下面通过图2.34来了解地球经纬度。

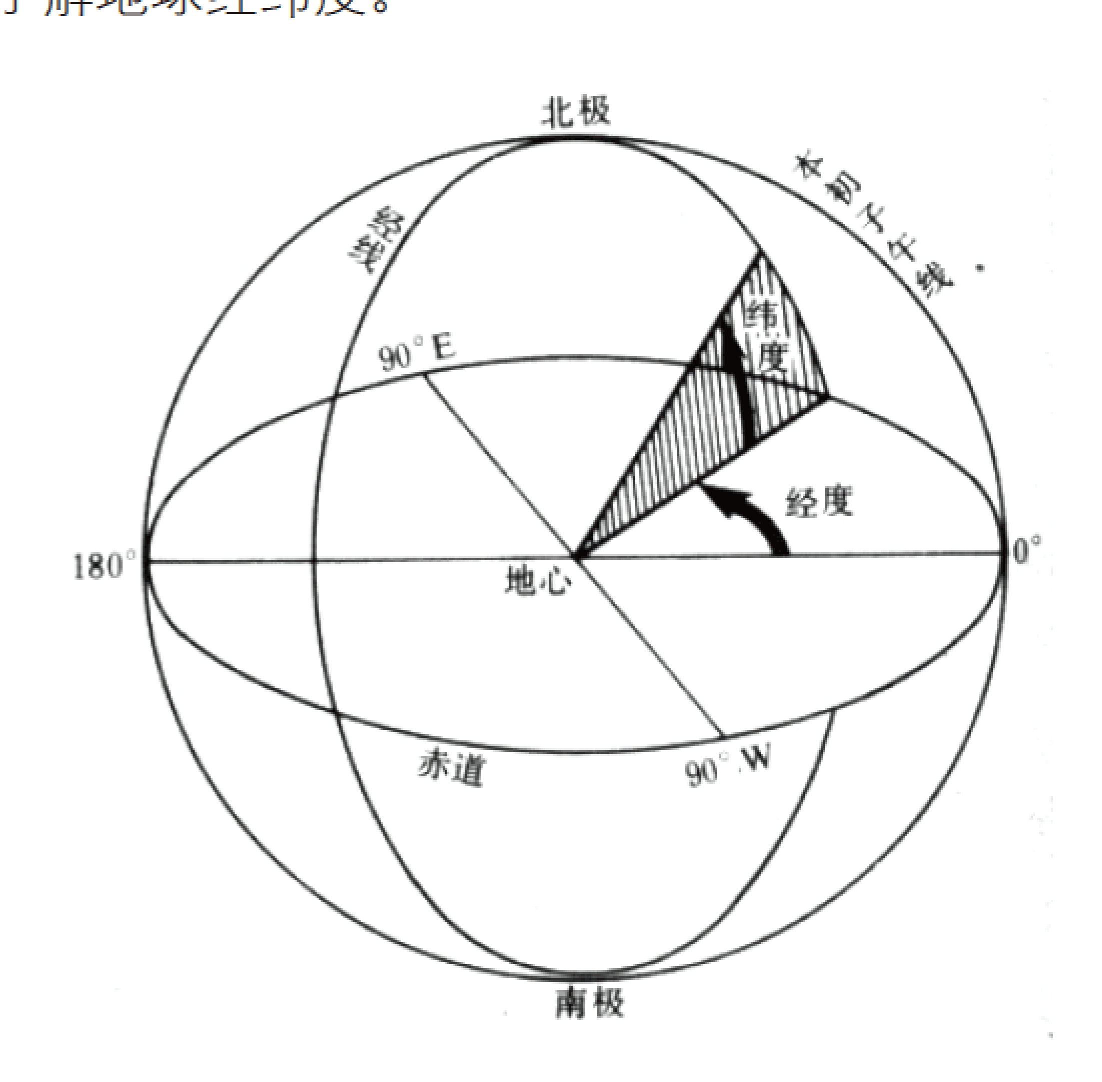


图2.34　地球经纬度

2.6.2 有哪些定位数据

HTML 5通过Geolocation接口获取用户地理位置信息，开发者不需要关心接口是在什么设备上、使用什么底层技术去实现的，只要会简单地调用即可。

一般来说，浏览器可以从设备中获取以下数据来源：

- IP地址。
- GPS（Global Positioning System，即全球定位系统）。
- RFID（Radio Frequency IDentification，即射频识别），如汽车防盗和无钥匙开门系统的应用、门禁和安全管理系统。
- Wi-Fi地址。

- GSM或CDMA手机的ID。
- 用户自定义的地理位置数据。

每种获取方式由于原理不同，所以在精准度上也会产生差异，比如使用笔记本连接Wi-Fi上网获取的经纬度信息与使用手机在GSM上获取的经纬度信息很可能会不完全一致。下面通过对比各项技术的优缺点让读者能够更加全面地了解差异，表2.16列出了定位数据来源的优缺点。

表2.16 定位数据来源优缺点

定位数据来源	优点	缺点
IP地址	连接上网的地方都可获取	不精确
GPS	非常精确	定位时间长、耗电大、室内效果差
RFID	精准、可在室内	接入设备少
Wi-Fi	精准、可在室内	需要有无线接入点
基于手机	较精准、可在室内	需要有手机网络，且基站点要多
用户自定义	自行输入位置更准备，定位快速	当用户位置发生变化时不准确

通过Geolocation HTML 5除了能获取到经纬度坐标外，还能提供位置坐标的精准度。对于某些较高级的硬件设备，浏览器通过Geolocation还能获取到海拔、海拔精准度、行驶方向和速度等，开发者可以通过该接口获取到与原生应用同样丰富的数据形式，开发出更多酷炫的功能，而这一切都可以在浏览器里实现。

2.6.3 怎么保护自己的隐私

地理位置信息涉及到用户的私隐，HTML 5 Geolocation设计之初就考虑到了这一点，除非用户明确允许，否则无法获取位置信息。

当用户访问一张使用HTML 5 Geolocation功能开发的页面时，浏览器会出现用户授权提示条，图2.35显示了在Chrome浏览器下用户授权条的样式。



图2.35 Chrome浏览器下Geolocation功能授权提示

不同的浏览器，Geolocation用户授权提示信息形式也不同，Firefox的授权提示如图2.36所示。



图2.36 Firefox浏览器下Geolocation功能授权提示

HTML 5 Geolocation方法在使用时除了会进行用户授权，浏览器还允许对过往进行授权的网站进行再修改，用户可以通过修改网站授权达到保护自己隐私的目的，比如在咖啡厅使用带有Geolocation的应用查找周边的商户信息，这时候应用通过经纬度信息定位周边商户，可以方便自己寻找信息，但当环境发生变化时，如回到自己的家中，此时可以重新将授权私有以起到保护作用。下面将通过图示来学习如何将授权的网站再次隐私。

以Chrome浏览器为例，首先，单击浏览器导航栏右侧圆形类似定位的按钮，如图2.37所示。



图2.37　单击Chrome浏览器导航栏右侧定位按钮

然后，单击弹出提示框的“管理位置设置”链接，此时会重新打开一个窗口，链接地址为“chrome://settings/contentExceptions#location”，新开的页面显示当前浏览器的地理位置信息情况列表，用户可以通过编辑列表选择是否再次对网站进行授权，如图2.38所示。

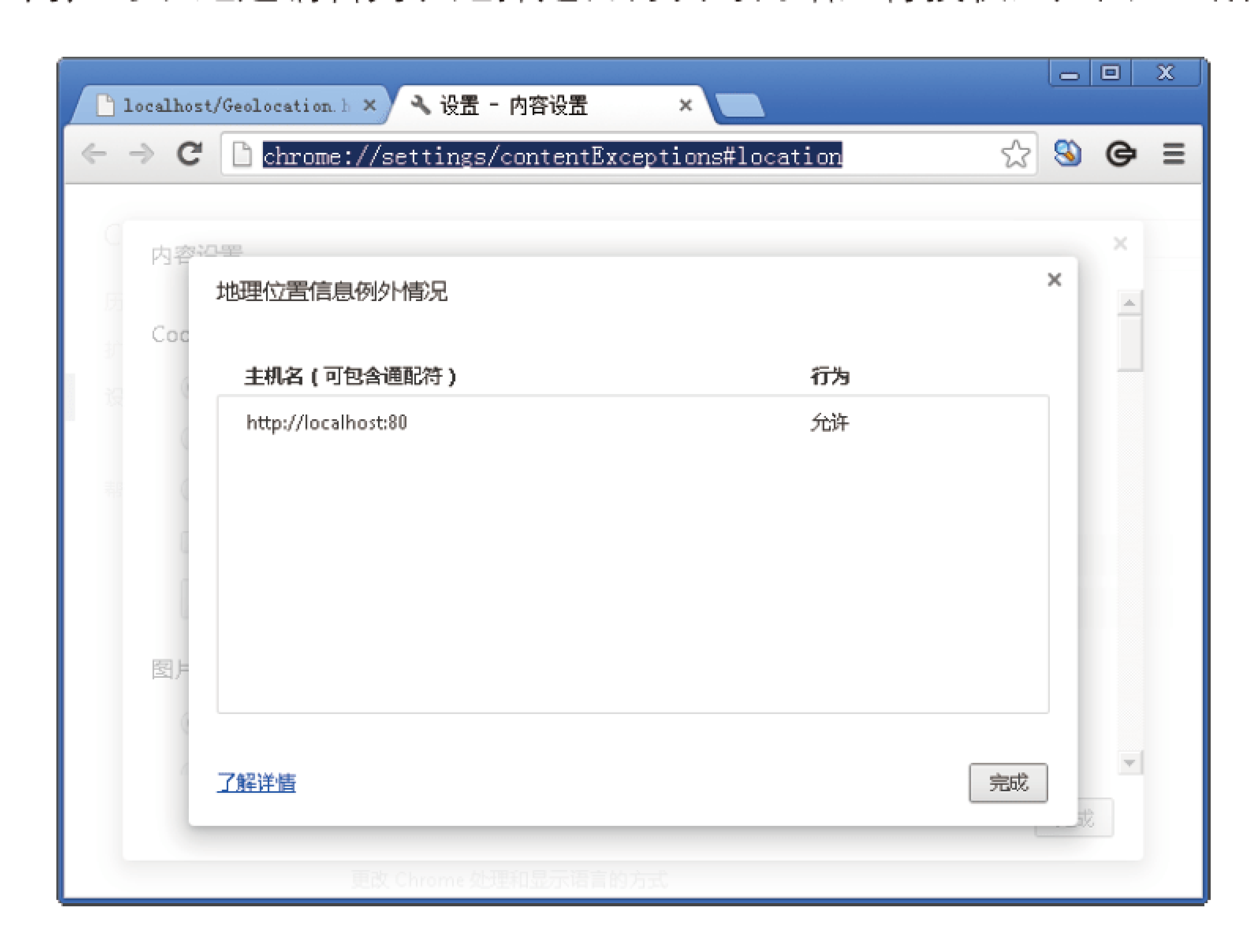


图2.38　Chrome浏览器Geolocation授权编辑列表

2.6.4　构建地理位置应用

在上一小节的示例应用中，已经通过HTML 5的Geolocation接口获取到当前用户的地理位置信息，下面将结合地图应用，在地图上显示用户当前的地理位置并进行标注，代码如下：

```
<!DOCTYPE html>
<html>
```

```
<script language="javascript" src="http://webapi.amap.com/maps?v=1.2"></script>          // 高德地图脚本
<body>
      <div id="imap" style="width:600px;height:200px;"></div>
      // 高德地图容器
</body>
<script>
window.onload = function(){
             // 页面资源加载完毕后执行
      navigator.geolocation.getCurrentPosition(function (position ) {
             var lnglat = new AMap.LngLat(position.coords.longitude,position.coords.latitude);                // 经纬度对象
             var mapObj = new AMap.Map("imap",{      // 实例化地图对象
                    center:lnglat,                   // 地图中心点
                    level:13                         // 地图缩放登记
             });
             var marker = new AMap.Marker({          // 实例化图标对象
                    map:mapObj,                      // 地图对象
                    position:lnglat,                 // 基点位置
                    icon:"http://webapi.amap.com/images/marker_sprite.png", // 图标，直接传递地址图片地址
                    offset:{x:-8,y:-34}              // 相对于基点的位置
             });
      }, function(error){
             debugger;
      }, {
             enableHighAccuracy : false,
             maximumAge : 10,
             timeout : 8000
      });
}
</script>
</html>
```

示例效果如图2.39所示。

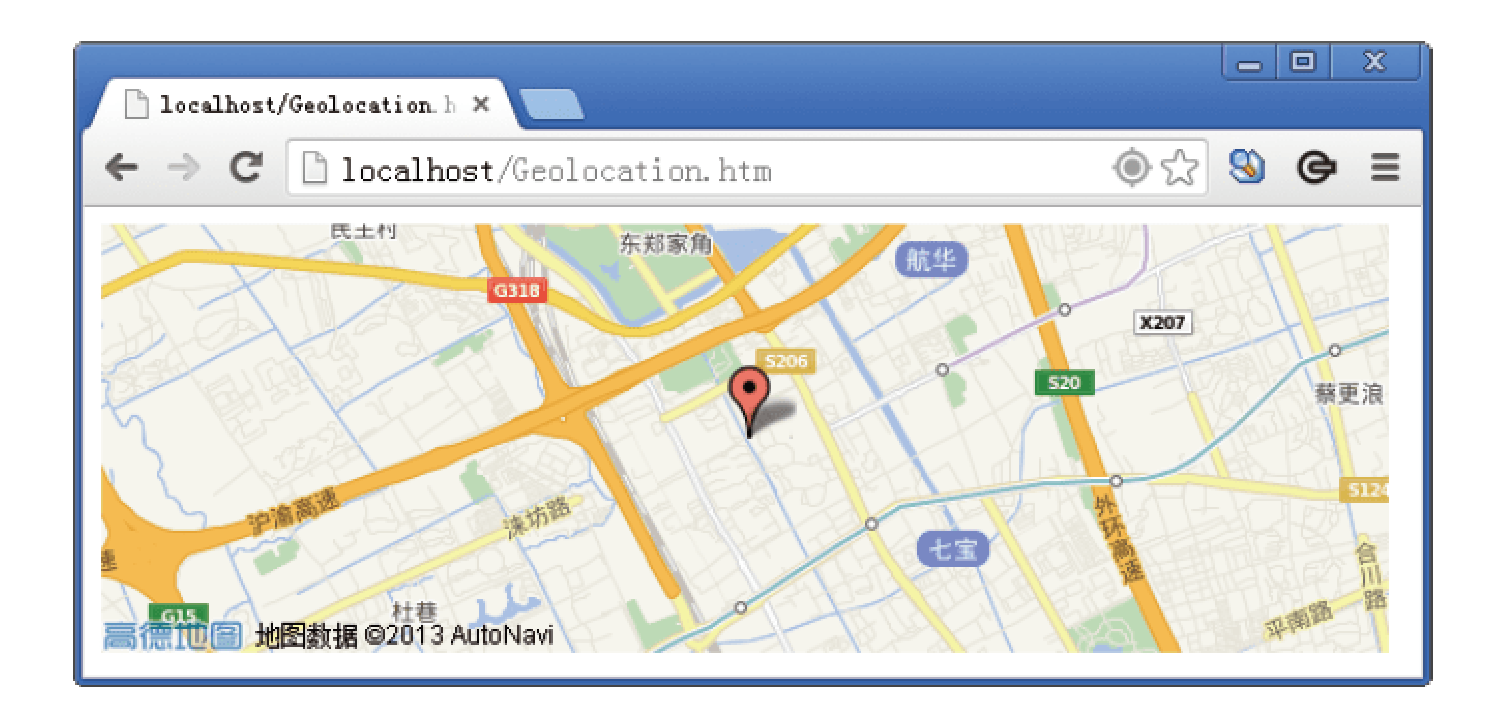


图2.39 通过Geolocation接口获取经纬度并显示在地图上

浏览器的Geolocation对象有三个方法，分析如下。

- getCurrentPosition：获取用户当前的位置信息，只能获取一次。
- watchPosition：循环检测用户的地理位置，只要发生变化，浏览器就会触发watchPosition函数。
- clearWatch：清除一个用于对用户位置的循环监视。

getCurrentPosition和watchPosition的用法类似，语法如下：

```
navigator.geolocation.getCurrentPosition(geolocationSuccess,
geolocationError, geolocationOptions);
```

geolocationSuccess当获取经纬度信息成功时触发，回调函数接收一个带有用户信息的对象字面量，包含两个属性coords和timestamp，其中coords属性对象包含以下7个属性值。

- accuracy：精确度。
- latitude：纬度。
- longitude：经度。
- altitude：海拔，海平面以上以米计。
- altitudeAcuracy：海拔的精确度。
- heading：朝向，从正北开始以度计。
- speed：速度，以米/每秒计。

geolocationError为错误回调函数，当无法获取用户经纬度时，浏览器会触发该函数，并传回错误对象，具体可能出现的错误情况可以参考文档http://dev.w3.org/geo/api/spec-source.html#permission_denied_error。

geolocationOptions参数为自定义的对象字面量，拥有三个自定义属性，分析如下。

- enableHighAccuracy：返回更加精确的用户信息数据，默认为false，关闭，如果设置为true，浏览器将消耗更多的时间用于获取信息，在移动设备上使用会消耗更多的电量。
- timeout：浏览器获取用户位置信息的超时时间，默认为0。
- maximumAge：浏览器获取用户位置信息后的缓存时间，单位为毫秒，默认为0，表示每次都重新获取。



2.7 拖放

拖放功能在HTML 5没有出现之前，可以通过元素的mousedown、mousemove、mouseup事件来实现，但仅针对于页面上的DOM元素。HTML 5的出现使得拖放变得更加广义，不仅另外提供了一套规范的事件格式，而且还支持桌面文件到浏览器的拖放，不需要依赖于第三方的插件，如最常用的Flash，大大简化了拖放的代码，提高了网页拖放的用户体验，同时也为移动设备拖放提供了一个可信赖的解决方案。

看到这里，读者一定对相关的拖放事件非常好奇，表2.17列出了跟拖放相关的事件。

表2.17 拖放相关的事件

事件名	说明
dragstart	开始拖放
drag	拖放过程中
dragenter	被拖放元素进入本元素范围内
dragleave	被拖放元素离开本元素范围
dragend	拖放结束
dragover	被拖放元素在本元素上移动
drop	放置拖放内容

开发者只要简单的通过上述事件，在不同的事件周期内执行指定业务逻辑，就可以轻松实现Web 2.0时代流行的拖放功能。

2.7.1 Datatransfer对象

Datatransfer对象是HTML 5新增的数据对象，主要用于控制拖放操作中的数据，提供了预定义的剪贴板格式数据访问功能，如可在dragstart事件触发时进行数据设置，之后在drop事件时对数据进行读取和修改。Datatransfer对象目前拥有如下4个属性。

- dropEffect：设置或获取拖曳操作的类型和要显示的光标类型，如类型none、copy、link和move。
- effectAllowed：设置或获取数据传送操作可应用于该对象的拖放效果，如效果none、copy、copyLink、copyMove、link、linkMove、move、all和uninitialized。
- files：拖放时的文件列表。
- types：返回在dragstart事件触发时为元素存储数据格式类型，如果是外部文件的拖放则返回files字符串。

另外，Datatransfer对象还拥有5种常用方法，说明如下。

- addElement：添加一同跟随拖曳的元素。
- clearData：删除指定格式数据，如未指设定，则删除当前元素下所有携带数据。
- getData：返回指定数据，如果数据不存在，则返回空字符串。
- setData：给元素添加指定数据。
- setDragImage：指定拖曳元素时，跟随鼠标移动的图片，x、y分别为相对于鼠标的偏移量。

Datatransfer对象一般在dragstart事件触发时使用，关键方法setData一般采用两种数据格式，用于文本数据存储的“text/plain”和用于URL信息存储的“text/uri-list”。

2.7.2 拖放的事件监听

拖放行为的所有监听事件如表2.18所示，HTML 5中要让元素具备拖放功能，首先需要打开元素draggable属性，设置为true，表示开启元素拖放功能，被设置的元素可以是网页上任意的图片、链接、文件或其他DOM节点。下面通过一个示例了解部分拖放事件的实际使用情

况，代码如下：

```
<!DOCTYPE html>
<html>
<style type="text/css">div{width:100px; height: 100px; background-color: blue;margin :5px;}</style>
<body>
      <div id="drag-target" data-content="我是数据" draggable="true">拽我！</div>    <!-- 可拖放元素 -->
      <div id="drat-dest"></div>                    <!--放置目标元素-->
</body>
<script type="text/javascript">
var   drag_target = document.getElementById('drag-target'),       // 可拖放元素
      drat_dest = document.getElementById('drat-dest');          // 可拖放元素
drag_target.addEventListener('dragstart', function(e) {          // 监听开始拖曳事件
      e.dataTransfer.setData('text/plain', e.target.getAttribute('data-content'));          // 使用文本存入属性值
},false);
drat_dest .addEventListener('drop', function(e) {          // 监听放置事件
      drat_dest.innerHTML = e.dataTransfer.getData('text/plain');       // 设置html为获取数据
},false);
drat_dest .addEventListener('dragover', function(e) {          // 监听拖曳移动事件
      e.preventDefault();                          // 阻止元素默认事件
},false)
</script>
</html>
```

将文件保存为html格式，并使用Chrome浏览器打开，效果如图2.40所示。单击第一区域方块，并拖放至第二区域，然后松开鼠标，此时第二区域出现“我是数据”文字，表示数据已经转移完毕，效果如图2.41所示。

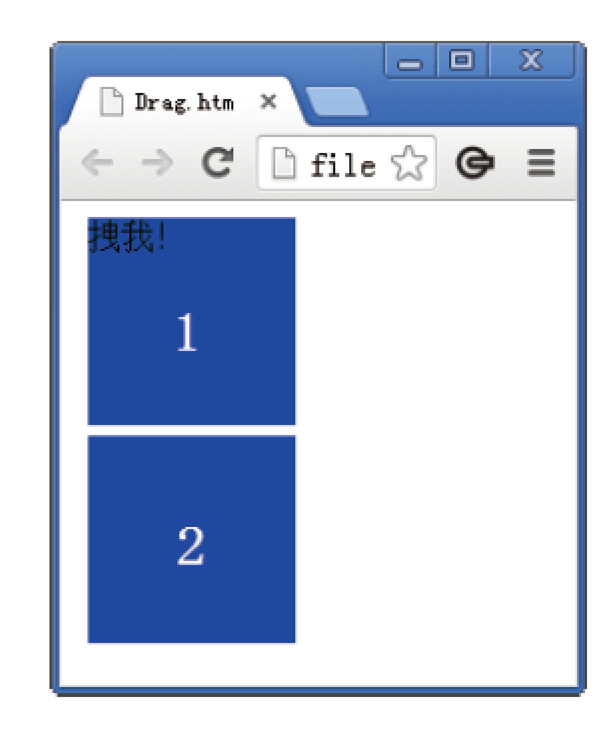


图2.40 拖曳示例页面效果



图2.41 拖放数据内容

整个示例中主要运用三种事件：dragstart、dragover、drop。dragstart事件监听拖放开始，示例中在拖放开始时，将第一区方块元素的数据使用DataTransfer对象的setData进行保存。dragover事件用于监听第二区方块，并调用事件对象的preventDefault方法，阻止浏览器元素默认事件，这个过程是必须的，否则无法触发drop事件。最后是监听drop事件，用于监听放置内容的结束过程，此时，从之前存入DataTransfer对象中获取数据并填充至第二方块区，示例代码非常简单，但完成了一个了不起的功能。

2.7.3 带拖放功能的网站

HTML 5拖曳功能的强大最先体现在上传功能的应用上，很多网站将其用于图片上传。大家所熟悉的新浪微博也与时俱进的提供了带有drag和drop功能的图片上传模块。

使用Chrome浏览器打开新浪微博，图片上传功能出现在发送微博区域，如图2.42所示。



图2.42 新浪微博图片上传功能

HTML 5的drop功能并非通过图2.42箭头所示的操作触发，而且出现在微博内容输入区域，实现了用户可直接将计算机本地的图片文件拖放至该区域并将图片上传。首先，在计算机本地用鼠标选中一张图片，将图片拖放至微博输入文本框区域，当正在拖曳图片的鼠标出现在文本输入区域时，网页提示用户上传准备就绪，效果如图2.43所示。



图2.43 拖曳本地图片至微博文本输入区域

松开拖曳图片的鼠标，图片文件被应用读取并上传至服务器，同时文本框下方出现图片缩略图，效果如图2.44所示。



图2.44 松开鼠标完成图片上传

微博应用的图片上传除了运用上一示例中出现的dragstart、dragover、drop外，还运用了dragenter、dragleave事件。以实现文件移入文本输入区域的信息提示和隐藏。

HTML 5的拖放功能除了出现在一些社交网站外，还出现在一些提供在线云存储的应用，如金山快盘等，为一些高级浏览器用户提供了更加优秀的用户体验。心动不如行动，赶快为自己的网站添加HTML 5拖放功能吧。

2.7.4 构建网页的拖放应用

体验了HTML 5拖放功能给网站应用带来的优秀用户体验，本节用制作示例来模拟微博图片拖放功能。打开光盘中的第2章示例Drop.htm，效果如图2.45所示。



图2.45 拖放示例

示例使用效果与微博相同，下面通过代码分析拖放功能的具体实现，示例代码如下：

```
<!DOCTYPE html>
<html>
<head><link rel="stylesheet" type="text/css" href="../css/weibo.css">
</head>
<body> /* 省略html结构代码，参看本书网络资源 */</body>
<script type="text/javascript">
var textarea = document.querySelector('textarea');   // 微博文本输入框元素
var W_layer = document.querySelector('div.W_layer'); // 图片上传提示层元素
var W_close = document.querySelector('a.W_close');   // 提示层关闭按钮元素
var W_add = W_layer.querySelector('li.add');         // 追加图片元素
var W_list = W_add.parentNode;                       // 图片列表容器元素
var p_count =document.querySelector('i.J_count');    // 图片数量元素
function render_one(file){                           // 单个图片元素渲染
      var reader;
// reader变量存放FileReader实例
      if (file.type.toLowerCase().match(/image.*/)) {
// 正则判断文件是否为图片类型
             reader = new FileReader();
// 实例化FileReader对象，用于读取文件数据
             reader.addEventListener('load', function (e) {
// 监听FileReader实例的load事件
                    var li = document.createElement('li');
// 创建上传图片容器
                    li.className = 'pic';
                    li.innerHTML ='<img src="' + e.target.result + '">';
                    W_list.insertBefore(li , W_add);         // 插入图片
                    p_count.innerHTML = (p_count.innerHTML>>0) + 1;
```

```
// 更新图片数
                });
                reader.readAsDataURL(file);                 // 读取文件为DataURL
        };
}
document.addEventListener('dragenter', function(e) {
// 监听拖放文件进入元素
        textarea.classList.add('textarea-enter');
},false);
document.addEventListener('dragover', function(e) {
// 监听拖放文件在元素上移动
        e.preventDefault();                    // 阻止默认事件，确保drop事件触发
},false);
document.addEventListener('dragleave', function(e) {
// 监听拖放文件离开元素
        textarea.classList.remove('textarea-enter');
},false);
document.addEventListener('drop', function(e) {      // 监听文件放置
        e.preventDefault();
        W_layer.style.display = 'block';
        textarea.classList.remove('textarea-enter');
        var files = e.dataTransfer.files;                // 获取文件列表
        for(var i =0,l = files.length; i<l; i++){
                render_one(files[i]);              // 页面构建每张上传图片
        };
},false);
W_close.addEventListener('click', function(e) {
// 监听浮层单击事件，执行关闭操作
        e.preventDefault();
        W_layer.style.display = 'none';
},false);
</script>
</html>
```

本示例主要实现的是监听HTML 5拖放事件，在不同的事件周期内执行对应操作，拖放相关涉及的事件有dragenter、dragover、dragleave、drop。

dragenter事件在每次文件被拖放进入元素范围内触发，示例中当文件被拖放入页面，则给文本框元素添加textarea-enter样式，提示用户支持将文件直接拖放上传到区域，如图2.43所示。

dragover事件当被拖放元素在本元素上移动时触发，要实现drop拖放功能，监听该事件在触发时阻止元素默认事件是必不可少的，否则无法在元素放置时触发drop事件。

dragleave事件在每次文件被拖放离开元素范围时触发，本例将该事件监听添加在document上，即表示当文件离开网页时触发，触发后移除原本被添加在文本框元素上的textarea-enter样式，表示此时放置元素将无法实现拖放功能，需再次拖放入网页。

drop事件是实现文件最终上传的关键，事件在文件被放置后触发，示例中当文件被放置于页面后触发drop事件，将从事件对象的dataTransfer属性中获取文件列表，并循环文件列表，渲染图片弹出上传提示信息框。

图片的渲染用到了自定义方法render_one，该方法接收一个File对象作为参数。首先通

过文件的type属性判断文件类型，当判断为图片文件时，表示该文件属于应用上传图片的范围，此时新建FileReader实例读取图片文件内容，并将返回的base64内容赋予图片，插入弹出的浮提示信息框。



本示例只完成了网页前端效果部分，不涉及后端图片保存功能，读者可在拖放章节的示例中学习如何使用Node.js编写图片上传接收服务。



2.8 Web存储

传统的客户端存储一直以来都是使用Cookies实现的，但限于Cookies自身的局限性，它并不适合存储大量数据。因为浏览器每次发送请求都会将Cookies信息发送至服务器端，这使得访问速度下降且不够高效，同时不同的浏览器对Cookie有不同的限制，如早期的IE 6浏览器对Cookies限制大小为5KB，同一域名下只允许存在20个Cookies，后续新增的Cookies数据会将先前的踢出，这将会导致很多意想不到的问题，所以使用Cookies一定要满足后台数据服务，同时要在知晓的情况下才可使用，切记滥用Cookies。

现今的Web应用需要更加强大的数据存储方式来替代早期的Cookies，在HTML 5出现之前已经有很多相关的技术解决方案，如IE的User Data、Flash的Cookie、谷歌的Gears技术等，每种技术都有自己的特点和局限，同时需要不同插件的支持，使得开发人员需要消耗大量的成本完成Cookies的替代方案，维护成本攀升。

HTML 5的出现，提供了全新的统一标准的Web存储解决方案，并且目前已经得到了广泛的支持，如IE 8+、Firefox 3.5+、Safari 4+、Chrome 4+、Opera 10.5+，手机平台包括iPhone 2+和Android 2+，都已经实现了并支持HTML 5的Web Storage。



W3C提供了Web Storage标准的说明文案，如果读者英文不错，可以先前往地址http://dev.w3.org/html5/webstorage/查阅。

2.8.1 设置和获取数据

HTML 5的Web存储提供了一套键值对的存储模型，同时对于不同网站，数据被存储于不同区域，并且一个域名只能访问自身的存储数据，在安全上做到了与Cookies同样的效果。存储大小也进行了调整，现在开发者可以使用HTML 5的Web存储至少5MB的数据，对于一般的应用来说已经绰绰有余。

HTML 5的Web存储一共提供了5种方法，说明如下。

- setItem(key,value)：添加存储键值对，存储数据为字符类型。
- getItem(key)：根据key值获取对应的值。
- clear()：清空Web存储中所有数据。

- removeItem(key)：从Web存储移除某个指定键值对应的数据。
- key(n)：获取第n个键值。

下面以LocalStorage功能介绍具体的方法使用，代码如下：

```
localStorage.setItem('1','test');
console.log(localStorage.getItem('1'))                    // 打印出数据'test'
```

读者可以使用Chrome浏览器打开开发者工具查看被存储的数据，效果如图2.46所示。

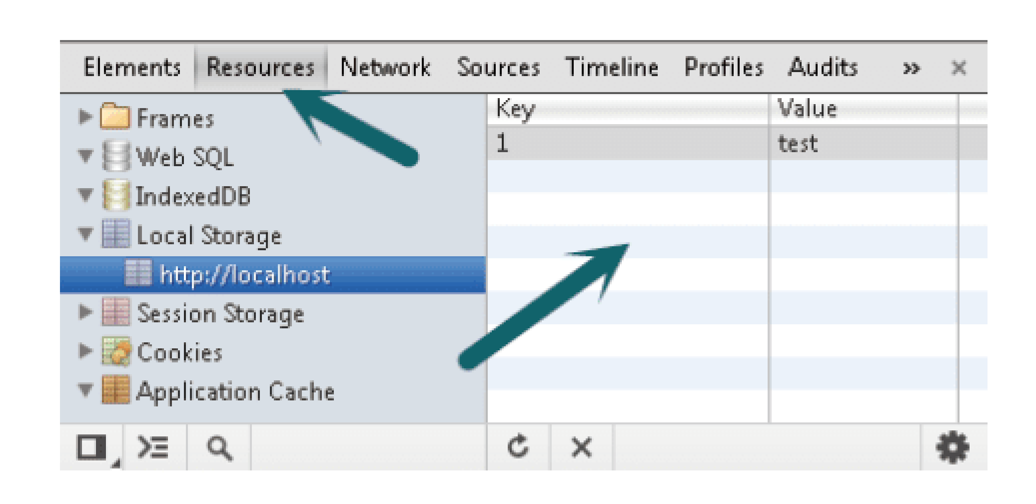


图2.46 使用Chrome浏览器查看localStorage数据

Web存储中的key方法，用于获取当前域名下对应索引序号的键值，游标默认从0开始，如果要获取之前存储的数据键值，代码如下：

```
localStorage.key(0);                               //  输出为'1'
```

想要删除一个或者多个Web存储数据可以使用removeItem或clear方法，代码如下：

```
localStorage. removeItem ('1');                    // 删除键值'1'对应的缓存数据
localStorage.clear();                              // 删除当前域名下所有缓存数据
```

2.8.2 LocalStorage与SessionStorage

HTML 5的Web存储提供了两种客户端数据存储方式，分别为LocalStorage和SessionStorage。LocalStorage主要用于持久化的本地存储，除非主动删除数据，否则数据永远不会过期。SessionStorage用于本地存储一个会话中的数据，这些数据在该会话结束时一同销毁，如浏览器关闭，所以SessionStorage不是一种持久化的本地数据。

SessionStorage功能在浏览器进程的内存中实现，浏览器关闭后自动销毁。LocalStorage功能是一种持久化的存储，数据被存储后实际被放置于用户计算机上，不同的浏览器实现的LocalStorage存放的格式和位置也不相同，表2.18例举了目前主流浏览器LocalStorage数据存放形式，以Windows系统为例。

表2.18 主流浏览器LocalStorage数据存放形式

浏览器	格式	加密方式	存放路径
Firefox	SQLite	明文	C:\Users\user\AppData\Roaming\Mozilla\Firefox\Profiles\tyraqe3f.default\webappsstore.sqlite
Chrome	SQLite	明文	C:\Users\user\AppData\Local\Google\Chrome\User Data\Default\Local Storage\
IE	XML	明文	C:\Users\user\AppData\Local\Microsoft\IE\DOMStore\
Safari	SQLite	明文	C:\Users\user\AppData\Local\Apple Computer\Safari\LocalStorage
Opera	XML	base64	C:\Users\user\AppData\Roaming\Opera\Opera\pstorage\



SQLite是一款轻型的数据库，遵守ACID的关联式数据库管理系统，主要被用于嵌入式产品，而且目前已经在很多嵌入式产品中使用，占用资源非常的低，在嵌入式设备中，可能只需要几百K的内存就够了。

下面将以Windows系统下Chrome浏览器为例，找寻用户本地的LocalStorage存储信息。首先在浏览器内打开一个网站（如http://www.dianping.com），在控制台输入测试存储的LocalStorage信息，代码如下：

```
localStorage.setItem('1','test');
```

代码执行效果如图2.47所示。



图2.47　储存测试LocalStorage信息

下载SQLite用于打开Chrome浏览器本地LocalStorage存储，地址为http://www.sqlite.org/download.html，下载链接如图2.48所示。

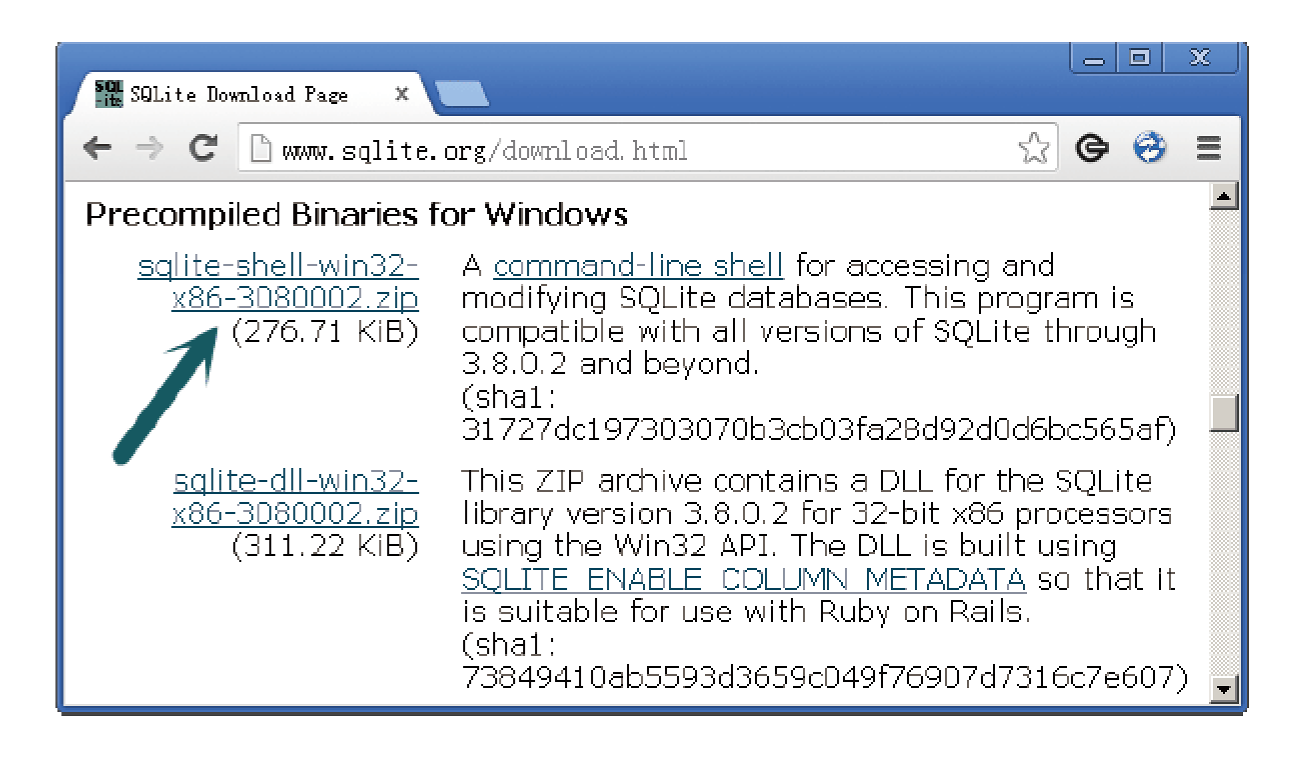


图2.48　下载SQLite

打开本地目录“C:\Users\{user}\AppData\Local\Google\Chrome\User Data\Default\Local Storage”，其中“{user}”按照读者本地用户而定，查找与“www.dianping.com”相关的文件，笔者计算机本地文件名为“http_www.dianping.com_0.localstorage”，使用SQLite打开，如图2.49所示。

```
C:\Windows\system32\cmd.exe - sqlite3.exe  "C:\Users\yao.zhou\AppData\Local\Goog...
D:\sqllite3>sqlite3.exe "C:\Users\yao.zhou\AppData\Local\Google\Chrome\User Data
\Default\Local Storage\http_www.dianping.com_0.localstorage"
SQLite version 3.7.10 2012-01-16 13:28:40
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .table
ItemTable
sqlite> select * from ItemTable;
dp-storage-local-expire|{}
footprint_guide|1
no-skin-tip|1
myinfo-tag|701
1|test
sqlite>
```

图2.49 使用SQLite打开本地LocalStorage存储文件

从中可以找到之前存入的数据，读者可以参照表2.18查找其他浏览器本地的LocalStorage信息，马上动手吧。

2.8.3 网站本地存储兼容性方案

HTML 5的LocalStorage功能目前只能在高级的浏览器上使用，如果要将LocalStorage用在网站上，需要考虑到历史浏览器的兼容问题，前面已经介绍使用Cookies在网络传输和浏览器上的瓶颈和局限，下面将介绍笔者编写的LocalStorage类库，用于解决不同浏览器的兼容问题，代码如下：

```
(function (WIN) {
    var NOOP = function () { },
    DOC = document,
    DB_NAME = 'html5_storage',                    // Web SQL数据库名
    DB_DISPLAYNAME = 'html5 storage data',        // Web SQL数据表名
    DB_MAXSIZE = 1048576,                         // Web SQL数据库大小
    DB_VERSION = '1.0',                           // Web SQL数据版本
    USERDATA_PATH = 'html5_storage',              // UserData路径
    USERDATA_NAME = 'data',                       // UserData存储键值
    MODE_NOOP = 0,                                // 无法匹配状态
    MODE_HTML5 = 1,                               // HTML 5状态
    MODE_GECKO = 2,                               // 部分Firefox状态
    MODE_DB = 3,                                  // 部分Safari状态
    MODE_USERDATA = 4;                            // 部分IE状态
    var data = {},                                // 缓存数据
    readys = [],                                  // 启动方法队列
    storageDriver,                                // 当前浏览器缓存
    storageMode;
    function _mix(r, s) {                         // 对象合并方法
        var key;
        for (key in s) { r[key] = s[key]; }
        return r;
    };
    try {                                         // 浏览器判断
        if (WIN.localStorage) {                   // 支持LocalStorage
            storageMode = MODE_HTML5;
        } else if (WIN.globalStorage) {// Firefox 2 and Firefox 3.0
```

```
            storageMode = MODE_GECKO;
        } else if (WIN.openDatabase && navigator.userAgent.indexOf('Chrome') === -1) {
            storageMode = MODE_DB;      // Safari 3.1 and Safari 3.2
        } else if (D.UA.ie >= 5) {      // IE 5、IE 6 and IE 7
            storageMode = MODE_USERDATA;
        } else {
            storageMode = MODE_NOOP;
        };
    } catch (ex) {
        storageMode = MODE_NOOP;
    };
    var Storage = {                             // 类库接口
        clear: NOOP,                            // 清空所有存储数据
        length: function () { return 0; },      // 获取缓存数量
        getItem: NOOP,                          // 根据键值获取数据
        removeItem: NOOP,                       // 移除单个键值对
        setItem: NOOP,                          // 存储键值对
        ready: function (fn) {                  // 类库模块准备完毕
            if (Storage.isReady) { fn();}
            else { readys.push(fn); };
        },
        isReady: false                   // 类库模块是否准备完毕
    };
    function _doReady() {                // IE 5、IE 6 and IE 7模块完毕
        DOC.body.appendChild(storageDriver);
        storageDriver.load(USERDATA_PATH);
        try { data = JSON.parse(storageDriver.getAttribute(USERDATA_NAME) || '{}');
        } catch (ex) { data = {}; };
        Storage.isReady = true;
    };                                   // 可以使用原生的LocalStorage方法
    if (storageMode === MODE_HTML5 || storageMode === MODE_GECKO) {
        _mix(Storage, {
            length: function () { return storageDriver.length; },
            removeItem: function (key) {
                storageDriver.removeItem(key);
            },
            setItem: function (key, value, json) {
                if (value === undefined || value === null) {
                    Storage.removeItem(key);
                } else {
                    storageDriver.setItem(key, json ? JSON.stringify(value) : value);
                };
            }
        });
        if (storageMode === MODE_HTML5) { // 可以使用原生的LocalStorage方法
            storageDriver = WIN.localStorage;
            _mix(Storage, {
```

```
                    clear: function () { storageDriver.clear(); },
                    getItem: function (key, json) {
                        try {
                            return json ? JSON.parse(storageDriver.getItem(key)) :
                                storageDriver.getItem(key);
                        } catch (ex) { return null; };
                    }
                });
            } else if (storageMode === MODE_GECKO) { // Safari 3.1 and 3.2
                storageDriver = WIN.globalStorage[WIN.location.hostname]; // 使用globalStorage方法
                _mix(Storage, {
                    clear: function () {
                        for (var key in storageDriver) {
                            if (storageDriver.hasOwnProperty(key)) { // 判断是否为非原型对象属性
                                storageDriver.removeItem(key);
                                delete storageDriver[key]; // 删除对应属性
                            };
                        };
                    },
                    getItem: function (key, json) { // 获取单条数据，支持JSON
                        try {
                            return json ? JSON.parse(storageDriver[key].value) :
                                storageDriver[key].value;
                        } catch (ex) { return null; };
                    }
                });
            };
            Storage.isReady = true;          // IE 5、IE 6 and IE 7下使用UserData
        } else if (storageMode === MODE_DB || storageMode === MODE_USERDATA) {
            _mix(Storage, {
                clear: function () {
                    data = {};
                    Storage._save();         // 调用私有方法存储
                },
                getItem: function (key, json) {
                    return data.hasOwnProperty(key) ? data[key] : null;
                },
                length: function () {        // 循环属性获取长度
                    var count = 0, key;
                    for (key in data) {
                        if (data.hasOwnProperty(key)) { count += 1; }
                    };
                    return count;
```

```
            },
            removeItem: function (key) {
                delete data[key];
                Storage._save();
            },
            setItem: function (key, value, json) {      // 存入当条数据，支持JSON
                if (value === undefined || value === null) {
                    Storage.removeItem(key);
                } else {
                    data[key] = value.toString();      // 转为字符型存储
                    Storage._save();
                };
            }
        });
        if (storageMode === MODE_DB) {         // 使用openDatabase方法
            storageDriver = WIN.openDatabase(DB_NAME, DB_VERSION, DB_DISPLAYNAME,DB_MAXSIZE);
            _mix(Storage, {
                _save: function () {
                    storageDriver.transaction(function (t) {      // 新建事务创建数据库
                        t.executeSql("REPLACE INTO " + DB_NAME + " (name, value) VALUES ('data', ?)", [JSON.stringify(data)]);
                    });
                }
            });
            storageDriver.transaction(function (t) {      // 新建事务创建数据表
                t.executeSql("CREATE TABLE IF NOT EXISTS " + DB_NAME + "(name TEXT PRIMARY KEY, value TEXT NOT NULL)");
                t.executeSql("SELECT value FROM " + DB_NAME + " WHERE name = 'data'", [], function (t,results) {
                    if (results.rows.length) {
                        try { data = JSON.parse(results.rows.item(0).value);
                        } catch (ex) { data = {}; };
                    };
                    Storage.isReady = true;
                });
            });
        } else if (storageMode === MODE_USERDATA) {      // IE 5、IE 6 and IE 7下使用UserData
            storageDriver = DOC.createElement('span');
            storageDriver.addBehavior('#default#userData');      // 添加存储数据行为
            _mix(Storage, {
                _save: function () {
                    var _data = JSON.stringify(data);
```

```
161                     try {
162                       storageDriver.setAttribute(USERDATA_NAME, _data);
163                       storageDriver.save(USERDATA_PATH);
    // 存入用户本地
164                     } catch (ex) { };
165                 }
166             });
167             if ($.isReady) { _doReady();
168             } else { $(document).ready(_doReady); };
    // 监听文档domready事件
169         };
170     } else { Storage.isReady = true; };
171     var interval = WIN.setInterval(function () {
    // 处理类库预准备完毕
172         if (Storage.isReady) {
173             readys.forEach(function (fn) { try { fn() } catch (e) { }; });
    // 循环执行监听方法
174             WIN.clearInterval(interval);
175             interval = null;
176             readys = [];
177         };
178     }, 100);
179     WIN.Storage = Storage;                    // 转到全局对象
180 })(this);
```



本类库依赖于JSON和jQuery，读者可以使用DouglasCrockford大师的JSON.js类库。

该类库支持市面上所有的浏览器，包括移动端浏览器，表2.19列举了不同兼容存储技术对应的浏览器类别。

表2.19 不同兼容存储技术对应的浏览器类别

存储技术	浏览器
LocalStorage	IE 8+、Firefox 3.5+、Safari 4+、Chrome 4+、Opera 10.5+、iPhone 2+、Android 2+
GlobalStorage	Firefox 2+、Firefox 2.x 、Firefox 3.0.x
Database Storage	Safari 3.1 、Safari 3.2
UserData	IE 5、IE 6、IE 7

代码第39行~第50行为类库的应用接口和属性，提供了6个方法和一个属性，说明如下。

- clear：清空当前域名下所有存储数据。
- length：获取当前域名下存储的键值对个数。
- getItem：获取当前域名某个键值的数值，传递的第三个参数为是否返回JSON格式。
- removeItem：移除当前域名下某个键值对。
- setItem：存储键值对到当前域名，传递的第三个参数表示是否存储为JSON格式。
- ready：类库的事件回调函数准备完毕。
- isReady：类库是否准备完毕。

与传统的LocalStorage不同，本类库的getItem和setItem方法允许接收JSON格式数据，而

传统的数据格式只能为字符串，这是考虑到日常开发的使用，而且还提供了length方法，方便开发者获取存储键值的个数。

2.8.4 如何在实际开发中使用本地存储

虽然HTML 5的Web存储技术非常强大，但并不能完全替代Cookies。Web存储是一种完全作用于客户端的存储技术，无法随着客户端请求被发送至服务器，而Cookies可以作为请求数据信息被发送至后端服务器进行处理，比如最常用的网页登录信息还是需要通过Cookies来做实现，但在前面的章节已经介绍过，过多的使用Cookies是一种低效甚至会引发一些意想不到的异常问题的方法，HTML 5的Web存储的出现正好补充了Cookies的这些局限。

当网站新上了一个产品，要让用户第一时间知道已经有一个新的功能可以使用，这时候需要制作一个提示框来提醒用户使用，当用户关闭提醒时只针对当前浏览器，而用户再次以其他浏览器打开时，提醒仍然出现。下面将结合上一章节的类库，通过示例制作这一功能，代码如下：

```
<!DOCTYPE html>
<html>
<style type="text/css">/* 样式代码省略，详见本书网络资源 */</style>
<script src="../js/jquery-1.8.3.js"></script>
<script src="../js/localstorage.js"></script>             <!-- localstorage类库引用 -->
<body>
<div class="top-tips" style="display:none">
      <p>网站有新功能上线啦，快来体验吧！</p>
      <a href="#" class="delt" title="不再提示">不再提示<span></span></a> <!-- 关闭按钮 -->
</div>
</body>
<script type="text/javascript">
      Storage.ready(function () {              // 存储模块准备就绪回调
             var tips = document.querySelector('div.top-tips'),      // 提示容器
                  key = 'html5_close',  // 存储键值
                  data = Storage.getItem(key);// 获取存储数据
             if(!data){
                  tips.style.display = 'block';
                  tips.querySelector('a.delt').addEventListener('click',function(e){// 监听关闭按钮单击事件
                          e.preventDefault();
                          Storage.setItem(key,1);             // 存储数据，表示已被用户关闭
                          tips.style.display = 'none';
                     },false);
             };
      })
</script>
</html>
```

使用Chrome浏览器打开示例，效果如图2.50所示。

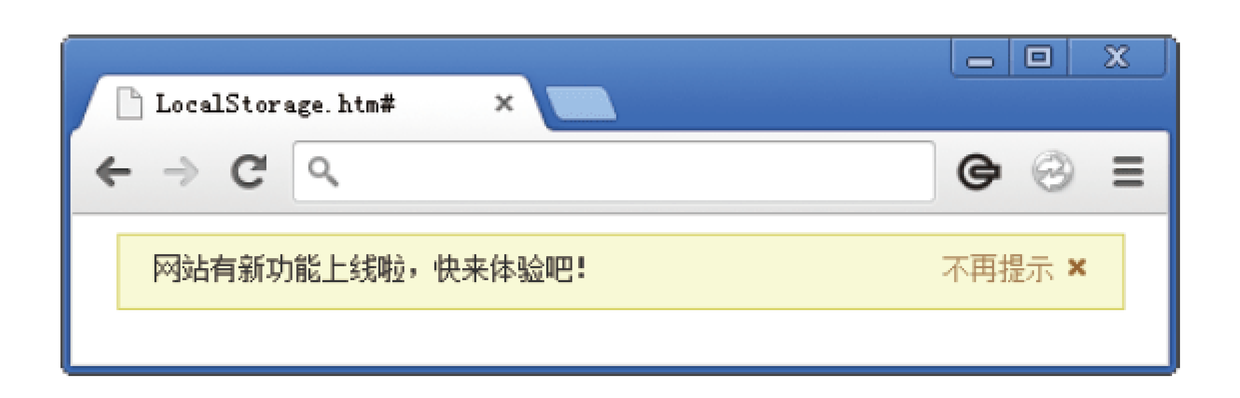


图2.50 LocalStorage示例效果

当用户单击右侧关闭按钮时，程序将调用LocalStorage类库的setItem方法，将数据1存储于键值“html5_close”中，下次打开网页时，首先会获取“html5_close”键值的数据，当发现数据存在时不显示提示信息。



2.9 HTML 5的通信

在HTML 5通信出现前的很长一段时间，网络的传输模式比较单一，都是围绕着HTTP的请求和响应模式而构建。由于HTTP的无状态性，浏览器加载完毕一张网页，然后直到用户操作进入下一页之前，都不会发生行为。

随着2005年Web 2.0的到来，Ajax技术开始兴起，原本静止的网页变得动态，但并没有改变所有HTTP通信都是由浏览器控制的局面。如果要实现数据定时更新，就需要使用Ajax做定期的轮询，以便从服务器加载最新的数据。后期由于部分产品实时通信的需要，也发展出了各种可以让服务器推送新数据的技术，例如最为著名的Comet技术，使用了Comet技术的产品如Google Talk。尽管出现了一些临时解决的技术方案，但有些如在线游戏的使用场景，HTTP请求所带来的额外消耗不容小觑，直接导致应用的延迟。HTML 5通信的出现解决了这些问题，其提供了更加标准和统一的解决方案，下面就将介绍各种HTML 5带来的通信技术。



Comet技术是一种用户Web的推送技术，读者可以参看维基百科了解Comet的相关信息，地址为http://zh.wikipedia.org/wiki/Comet_(web技术)。

2.9.1 PostMessage API

PostMessage又被称为跨文档消息通信。浏览器在设计之初，出于安全的考虑，在同一个浏览器中的框架、标签页、窗口之间的通信一直受到严格的安全限制，但实际又存在一些让不同站点之间通信交互的需求，因此HTML 5在规范中添加了跨文档消息通信，该规范可以确保标签页、窗口、IFrame之间进行跨源通信，统一使用PostMessage定义的应用程序接口发送数据。



所谓同源，就是指域名、协议、端口相同，是由Netscape提出的一个著名的安全策略，现在所有的可支持JavaScript的浏览器都会使用这个策略，其实就是所谓的“跨域”问题。

目前各浏览器对HTML 5的PostMessage接口支持情况不同，表2.20列举了目前主流浏览器的支持情况。

表2.20 主流浏览器PostMessage API支持情况

浏览器	支持版本
Chrome	2.0 +
Firefox	3.0+
IE	8.0+
Opera	9.6+
Safari	4.0+

PostMessage功能最常被用在与IFrame之间通信，下面通过一个示例介绍该功能的使用，示例功能需要两张页面完成，主页面代码如下：

```
<!DOCTYPE html>
<html>
<style type="text/css">
  section{
    border-radius: 10px; background-color: #f4f4f4;
    height: 200px; width: 400px; margin: 0 auto; padding: 10px;
  }
  iframe{ margin-top: 5px; width: 395px; height: 160px; }
<!-- 窗口样式 -->
</style>
<body>
  <section>
    <div>发送信息<input></input><button>发送</button></div>
<!--  输入、发送区 -->
    <div>目标接收窗口</div>
    <iframe src="http://html5.me/PostMessage-Host.htm"></iframe>
<!-- 窗口页面引用 -->
  </section>
</body>
<script type="text/javascript">
  var iframe_win = document.querySelector("iframe").contentWindow;
// 窗口容器内部window对象
  document.querySelector('button').addEventListener('click',function(e) {
// 监听发送按钮单击事件
    e.preventDefault();                // 阻止默认提交事件
    iframe_win.postMessage(            // 向窗口发送信息
      document.querySelector("input").value,         // 发送内容
      "http://html5.me"                // 目标数据源地址
    );
  })
</script>
</html>
```

窗口子页面主要负责接收信息做展示，并判断传入数据的安全性，功能代码如下：

```
<!DOCTYPE html>
<html>
<body>
  <header>接收信息：</header><ul></ul>          <!-- 数据接收展示区 -->
</body>
<script>
window.onmessage = function(e){                  // 监听postmessage接收
  if ( e.origin !== "http://localhost" ) {       // 判断发送端源地址
    return;
  };
  var li = document.createElement('li');         // 创建数据条目
  li.innerHTML = e.origin + " 发送：" + e.data;
  document.querySelector('ul').appendChild(li);        // 展示数据
};
</script>
</html>
```

运行该示例首先需要布置两张页面，以Windows操作系统为例，将主页面和子页面均布置在Apahce搭建的Web服务器上（或使用IIS、Nginx等）。示例中的场景需要主页面和子页面分属于两个不用的域名，本示例主页面域名为http://localhost，子页面的域名为http://html5.me，子页面域名是一个修改了本地host文件的伪造域名，可以用于通过修改本地计算机目录C:\Windows\System32\drivers\etc下的hosts文件，在其中添加一行代码实现：

```
127.0.0.1     html5.me
```

准备就绪后，使用Chrome浏览器打开网址，效果如图2.51所示。

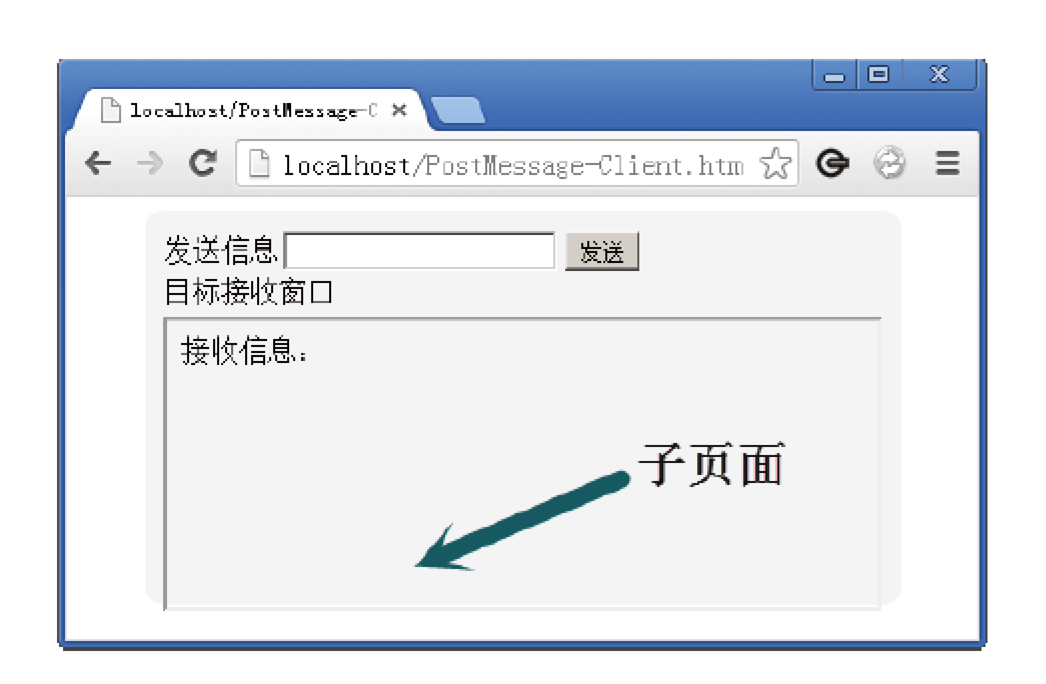


图2.51 PostMessage主页面效果

在主页面输入框内随意输入123字符串，单击“发送”按钮，子页面接收主页面传递的内容并显示在页面上，效果如图2.52所示。

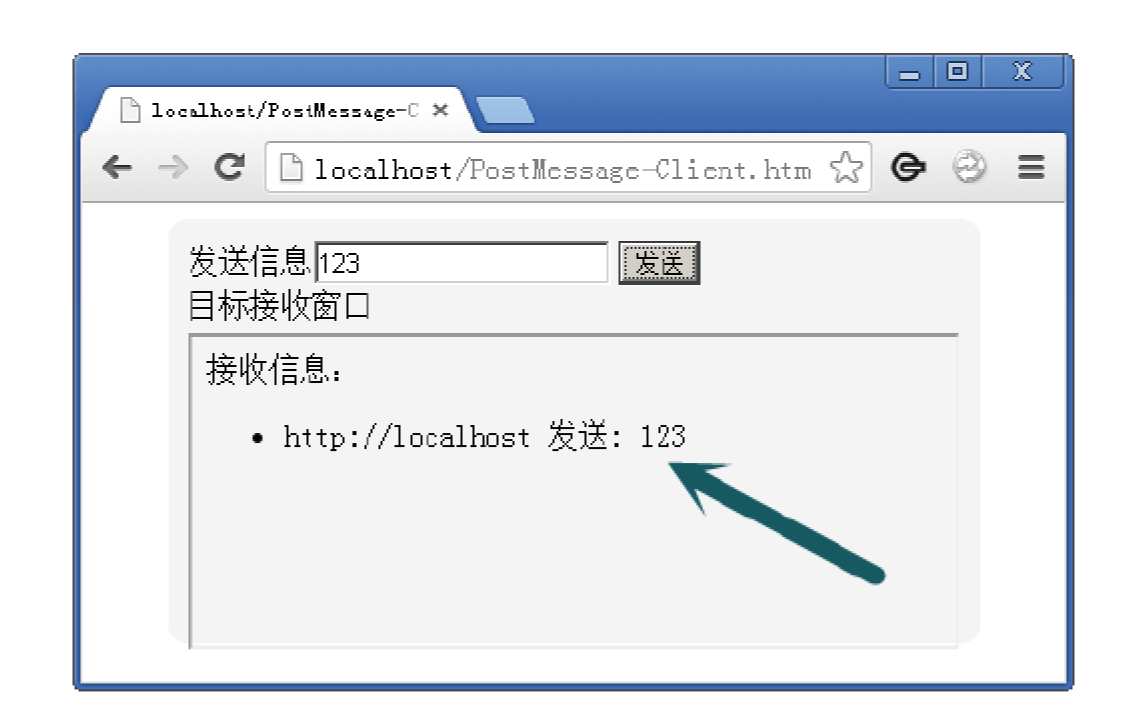


图2.52　主页面输入发送内容，子页面即时显示

示例关键方法是postMessage，接收两个参数，第一个参数表示发送的数据内容，第二个参数表示消息传送目标页面的域名地址。在目标页面上通过监听window的message事件接收主页面传递过来的数据，同时利用事件回调函数返回的事件对象属性origin判断过滤数据源，最后通过事件对象属性data获取主页面传送的数据。

2.9.2 XMLHttpRequest Level 2

XMLHttpRequest Level 2是早期XMLHttpRequest的升级版，最初的XMLHttpRequest被设计为仅限于同源通信，这使得同一站点的二级、三级域名之间传输数据无法实现，更不用说不同站点域名之间的数据传输，导致后续演变出了JSONP方法。

XMLHttpRequest Level 2对同源策略进行了修改，允许实现跨源的数据请求，同时还添加了progress事件用于监听请求进度，返回进度信息。XMLHttpRequest Level 2要求所有跨域的请求都要使用HTTP协议中的origin信息头，同时数据接收服务器需要具备CORS策略，各种服务器的CORS策略设置可以参考网站http://enable-cors.org/。



JSONP利用script标签里面的跨域特性进行跨域数据访问，在script标签里面存在的是一个跨域的URL，实际执行的时候通过这个URL获得一段字符串，这段返回的字符串必须是一个合法的JavaScript调用，通过EVAL方法执行这个字符串来完成对获得的数据的处理。

目前主流浏览器对XMLHttpRequest Level 2的支持情况如表2.21所示。

表2.21　主流浏览器XMLHttpRequest Level 2支持情况

浏览器	支持版本
Chrome	2.0+
Firefox	3.5+
IE	10.0+
Opera	12.0+
Safari	4.0+

下面通过示例介绍XMLHttpRequest Level 2的使用，示例分为两部分，客户端和服务器端，服务器端采用Node.js技术开发。

浏览器客户端代码如下：

```
<!DOCTYPE html>
<html>
<body>
  数据：<input></input><button>获取</button>    <!-- 数据获取显示区 -->
  <script>
  document.querySelector('button').addEventListener('click',function(e){
      // 监听按钮单击事件
    e.preventDefault();                          // 阻止按钮提交默认提交事件
    var xhr = new XMLHttpRequest();              // 实例XMLHttpRequest对象
    if(typeof xhr.withCredentials === undefined){
      // 判读浏览器是否支持Level 2
      console.log("浏览器不支持html5 XMLHttpRequest Level 2的跨源请求支持");
    }else{
      xhr.onload = function(){                   // 监听ajax onload事件
        var data =JSON.parse(xhr.responseText)         // 将文本转为JSON数据
        document.querySelector('input').value = data.data;
// 显示返回数据
      };
      xhr.onerror = function(e){ console.log(e) };  // 监听Ajax 错误事件
      xhr.open("GET", "http://localhost:8080/", true);
      // 设置请求地址和方法
      xhr.send();                                // 发送Ajax
    };
  });
  </script>
</body>
</html>
```

服务器端采用Node.js技术编写，代码如下：

```
var http = require("http");              // 引用http模块，用于Web服务器
http.createServer(function (req, res) {         // 创建新服务器
    res.setHeader('Access-Control-Allow-Origin', 'http://localhost');
// 所有域名跨域访问均可以被通过
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST');
// 服务器支持 ' GET, POST ' 方法
    req.setEncoding('utf8');             // 设置接收数据编码格式为 UTF-8
    res.end(JSON.stringify({data:'Hello World!'}));  // 返回测试数据
}).listen(8080, function () {            // 设置Web服务器监听端口，并启动服务
    console.log('listening on http://localhost:8080');
// 控制台显示Web服务器启动成功
});
```

首先启动Node.js服务，在Windows下打开命令行进入服务对应的“XMLHttpRequestLevel2-Server”文件夹，执行代码如下：

```
node server.js
```

如果启动成功，命令行提示如下：

```
listening on http://localhost:8080
```

将浏览器客户端页面部署在Web服务器上，如IIS、Apache、Nginx等。打开示例页面，

效果如图2.53所示。



图2.53 XMLHttpRequest Level 2客户端页面效果题

单击“获取”按钮，页面通过XMLHttpRequest Level 2请求远端非同源地址“http://localhost:8080/”，获取的数据显示在文本框内，效果如图2.54所示。



图2.54 单击“获取”按钮从远端非同源服务器获取数据

在判断浏览器是否支持XMLHttpRequest Level 2情况时，可以通过判断对应实例对象的withCredentials属性是否存在，如果存在则表明浏览器支持XMLHttpRequest Level 2，其他的用法与传统的XMLHttpRequest相同。

XMLHttpRequest Level 2对远端请求的服务器有一定的要求，需要在服务器端返回的HTTP信息头中指明支持的域名，示例中使用Node.js设置的Access-Control-Allow-Origin信息头为“http://localhost”，表示支持非同源的“http://localhost”发送的客户端请求，如果想对所有请求都做通过处理，可以设置为“*”号。如果要指定多个，一般会认为使用以下写法：

```
res.setHeader('Access-Control-Allow-Origin', 'http://localhost,
http://test.com');                          // 域名之间用逗号
res.setHeader('Access-Control-Allow-Origin', 'http://localhost
http://test.com ');                         // 域名之间用空格
```

非常不幸的是，浏览器无法识别上述两种返回头，推荐的使用方法是先对请求域名做判断，如果符合业务要求，再设置对应域名的Access-Control-Allow-Origin头信息。

2.9.3 WebSocket API

WebSocket的出现，使得服务器与客户端在允许的时间范围内互相推送信息成为可能。传统的Ajax技术需要由客户端主动发起请求，而WebSocket可以在服务器与客户端之间彼此相互推送，同时还允许跨域通信。越来越多的对即时性要求很强的应用也可以通过WebSocket完美解决，如股票信息更新显示、实时的网页聊天、多人在线游戏等。

WebSocket的通讯流程可以通过时序图2.55表示。

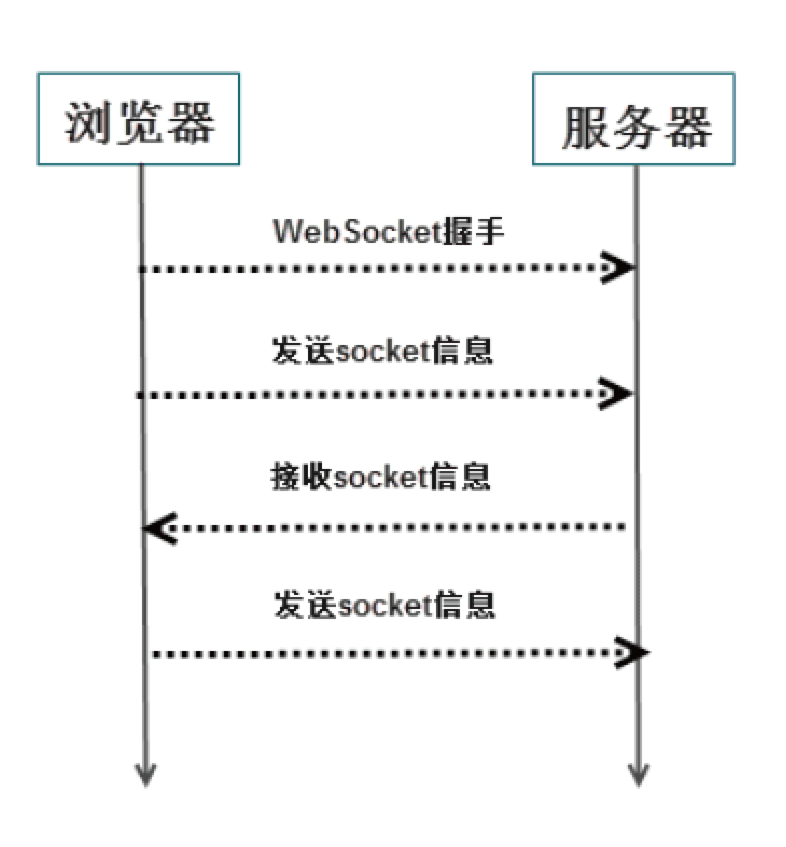


图2.55 WebSocket的通讯流程

WebSocket目前已经得到众多主流浏览器的支持，表2.22显示了目前浏览器的支持情况。

表2.22 主流浏览器WebSocket支持情况

浏览器	支持版本
Chrome	4.0 +
Firefox	4.0+
IE	10.0+
Opera	10.7+
Safari	5.0+

WebSocket的使用还是相对简单的，通过WebSocket构造函数就能创建WebSocket链接，代码如下：

```
var socket = new WebSocket('ws://html5.me', ['soap', 'wamp']);
```

以ws开头的协议为WebSocket新增的连接协议，另外对于安全的WebSocket连接有wss协议，类似于大家熟悉的https协议为http的安全协议。

构造函数的第二个参数表示可接收的子协议，允许传递单个字符串或者由字符串组成的数组，除了示例代码中使用的soap和wamp外，目前WebSocket还接收其他多种子协议，详情可以参考网站http://www.iana.org/assignments/websocket/websocket.xml。

WebSocket提供了多种事件用于监听数据传输，如下：

```
socket.onopen = function () {                        // WebSocket请求开始
  socket.send('客户端数据');                          // 向服务器发送数据
};
socket.onerror = function (error) {
  console.log('WebSocket 错误: ' + error);           // WebSocket错误监听
};
socket.onmessage = function (e) {                    // WebSocket返回数据监听
  console.log('服务器返回数据: ' + e.data);
};
socket.onclose = function (e) {                      // WebSocket请求关闭监听
  console.log('通道关闭');
```

```
    };
```



更多WebSocket标准介绍可以参考网站http://www.w3.org/TR/2012/WD-websockets-20120524/。

2.9.4 Socket.IO通信框架介绍

Socket.IO是Guillermo Rauch开发的基于Node.js的应用项目，以实现跨浏览器和跨平台应用为目标。Guillermo Rauch本人是LearnBoost公司的首席技术官以及LearnBoost实验室的首席科学家。Socket.IO针对于不同的浏览器会做自动优雅降级，选择当前浏览器最合适的实现方式，如在一些不支持HTML 5 WebSocket的浏览器上，会使用长连接的Ajax技术。同时Socket.IO提供了一套平台统一的应用程序接口，开发者在使用时完全不需要考虑浏览器的兼容问题，只需要把注意力放在业务开发上。



Socket.IO官网地址为http://socket.io/。

在使用Socket.IO功能时，首先需要使用Node.js的包管理工具下载Socket.IO和其相关的依赖模块，执行代码如下：

```
npm install socket.io
```

在示例文件夹执行后，包管理工具会自动下载相关依赖，如图2.56所示。

```
E:\SocketIO>npm install socket.io
npm http GET https://registry.npmjs.org/socket.io
npm http 200 https://registry.npmjs.org/socket.io
npm http GET https://registry.npmjs.org/socket.io/-/socket.io-0.9.16.tgz
npm http 200 https://registry.npmjs.org/socket.io/-/socket.io-0.9.16.tgz
npm http GET https://registry.npmjs.org/socket.io-client/0.9.16
npm http GET https://registry.npmjs.org/base64id/0.1.0
npm http GET https://registry.npmjs.org/redis/0.7.3
npm http GET https://registry.npmjs.org/policyfile/0.0.4
npm http 200 https://registry.npmjs.org/socket.io-client/0.9.16
npm http GET https://registry.npmjs.org/socket.io-client/-/socket.io-client-0.9.
16.tgz
npm http 200 https://registry.npmjs.org/redis/0.7.3
npm http GET https://registry.npmjs.org/redis/-/redis-0.7.3.tgz
npm http 200 https://registry.npmjs.org/policyfile/0.0.4
npm http GET https://registry.npmjs.org/policyfile/-/policyfile-0.0.4.tgz
npm http 200 https://registry.npmjs.org/base64id/0.1.0
npm http GET https://registry.npmjs.org/base64id/-/base64id-0.1.0.tgz
npm http 200 https://registry.npmjs.org/socket.io-client/-/socket.io-client-0.9.
16.tgz
npm http 200 https://registry.npmjs.org/redis/-/redis-0.7.3.tgz
npm http 200 https://registry.npmjs.org/policyfile/-/policyfile-0.0.4.tgz
npm http 200 https://registry.npmjs.org/base64id/-/base64id-0.1.0.tgz
```

图2.56 Socket.IO安装下载

下面的示例展示了一个简单的与服务器双向通信的过程，客户端代码如下：

```
<!DOCTYPE html><html>
<body>
显示：<input></input><br>                              <!-- 信息显示区域 -->
类型：<select>                                         <!-- 获取数据类型选择 -->
        <option value="year">年</option>
        <option value="month">月</option>
        <option value="day">日</option>
```

```
        </select>
        <button>获取</button>                    <!-- 获取操作按钮 -->
    <script src="/socket.io/socket.io.js"></script>          <!-- 通信兼容脚本 -->
    <script>
    var socket = io.connect('http://localhost:8080');        // 与服务器连接
    socket.on('news', function (data) {           // 监听服务器自定义事件
      document.querySelector('input').value = data;          // 将服务端返回的数据输出显示
    });
    document.querySelector('button').addEventListener('click',function(e){        // 监听获取单击事件
      e.preventDefault();                         // 阻止按钮默认提交事件
      socket.emit('test', document.querySelector('select').value);        // 向服务器发送信息
    });
  </script>
  </body></html>
```

服务端采用Node.js搭建，利用了Socket.IO通信框架，代码如下：

```
  var app = require('http').createServer(handler)       // 创建服务器模块
    , io = require('socket.io').listen(app)      // socket.io监听服务器端口
    , fs = require('fs');                        // 引用文件读取模块
  app.listen(8080);                              // 启动服务器监听8080端口
  function handler (req, res) {
    fs.readFile(__dirname + '/SocketIO.htm',     // 服务根目录文件
    function (err, data) {                       // 读取回调事件
      if (err) {
        res.writeHead(500);
        return res.end('页面 SocketIO.htm 加载失败');  // 失败返回页面错误信息
      }
      res.writeHead(200, {'Content-Type': 'text/html'});          // 设置成HTTP信息头
      res.end(data);                             // 返回页面数据
    });
  }
  io.sockets.on('connection', function (socket) {// 监听socket连接事件
    socket.emit('news', '欢迎加入WebSocket' );   // 向连接客户端广播数据
    socket.on('test', function (data) {          // 监听自定义test事件
      switch(data){                              // 判断获取的数据类型
        case 'year' : socket.emit('news', new Date().getFullYear());break;          // 返回当前年信息
        case 'month' : socket.emit('news', new Date().getMonth()+1);break;          // 返回当前月信息
        case 'day' : socket.emit('news', new Date().getDate());break;          // 返回当前日信息
      }
    });
```

```
25  })
```

进入示例代码的SocketIO文件夹，启动Node.js服务，代码如下：

```
node server.js
```

打开浏览器，在地址栏输入页面地址http://localhost:8080/，效果如图2.57所示。页面打开后，浏览器与服务器通过WebSocket连接，服务端向客户端发送欢迎信息，客户端接收后显示在文本框内。单击“获取”按钮，此时浏览器客户端向后端发送获取请求，服务器对请求的传入数据类别做出判断，返回相应的数据，客户端接收数据显示在文本框内，效果如图2.58所示。



图2.57　客户端浏览效果

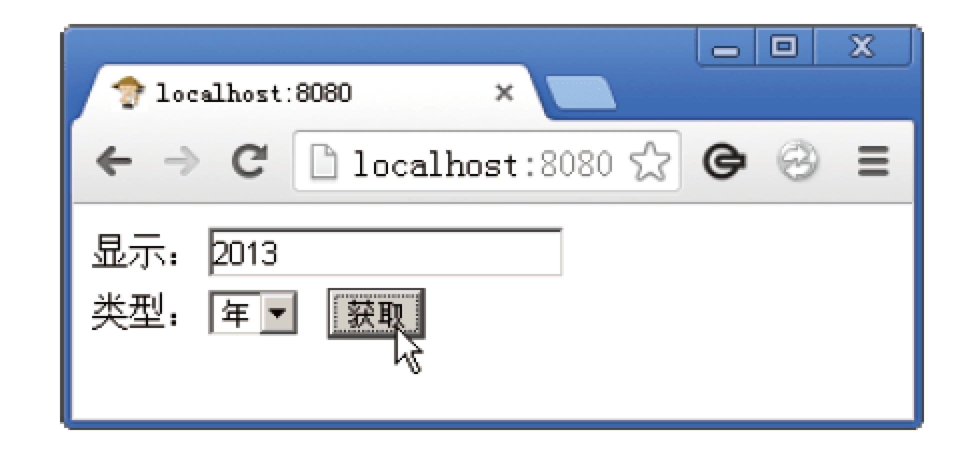


图2.58　单击页面的“获取”按钮

通过示例可以发现，使用WebSocket完全没有任何延迟，服务器端可以同时支持多个客户端接入，并可以同时推送和广播多个接入客户端。Socket.IO为开发者抹平了浏览器兼容，是Node.js下实时应用开发的最好选择，赶快动手写个自己的示例，一同感受WebSocket世界的精彩吧。

2.10 Web Workers

浏览器中的JavaScript一直运行在单线程的环境中，所以在做网站前端优化时，习惯性的将JavaScript放置在页面的底部，这样可以让页面的内容更快的加载渲染。有的时候，遇到前端处理页面大量数据会使用setTimeout方法，错开任务的执行时间，避开同一时间执行造成的交互停滞。这些做法虽然在一定程度上能起到优化效果，但显而易见的是浏览器单线程执行JavaScript制约了浏览器实现富应用的脚步。

HTML 5提出了Web Workers新功能，使得JavaScript可以类似于线程一样的实现并行，充分利用现在计算机多核的处理优势。开发者可将一些脚本处理的计算密集型任务分配给Web Workers处理，而不阻碍页面UI与用户的交互。

Web Workers虽然强大，但也需要在合适的场景内使用，对于有些情况的处理还是受到浏览器的限制，比如在Web Workers中执行的脚本无法访问页面window对象，也就是说，Web Workers无法直接访问页面与DOM相关的程序接口。同时，如果开发过程中出现过多的实例化Web Workers对象处理数据，会对计算机的CPU产生消耗，导致系统响应速度降低。

所以，只有在合适的场景使用Web Workers可以使浏览器应用变得更加流畅，目前，市面上的浏览器对Web Workers的支持情况各不相同，表2.23为主流浏览器对Web Workers的支持情况。

表2.23 主流浏览器对Web Workers的支持情况

浏览器	支持版本
Chrome	4.0+
Firefox	3.5+
IE	10+
Opera	10.6+
Safari	4.0+

接下来，将了解Web Workers的基本使用，同时介绍如何与页面进行通信，最后会演示一个小示例学习在实际开发过程中使用Web Workers。

2.10.1 与HTML5 Web Workers通信

作为一项在浏览器内使用的新技术，与页面DOM交互是必不可少的，但是Web Workers不能直接对DOM进行处理，需要通过事件模型和postMessage方法实现。postMessage接受字符串或JSON对象作为参数，具体使用情况取决于用户的浏览器，在新型的浏览器中可以支持传递JSON对象。

下面通过一个Hello World示例介绍Web Workers的使用，浏览器端代码如下：

```
var worker = new Worker('web_worker.js');  // 实例化Web Workers对象，传递脚本
worker.addEventListener('message', function(e) {
// 监听worker 的message事件，接收返回信息
  console.log(e.data);                     // 打印返回数据
}, false);
worker.postMessage('Hello World');         // 向Web Workers示例发送信息
```

Web Workers脚本文件web_worker.js的代码如下：

```
self.addEventListener('message', function(e) {
// 监听message事件，接收页面传递数据
  self.postMessage(e.data);                // 向调用页面输送信息
}, false);
```

2.10.2 多个JavaScript文件的加载与执行

对于一些较为复杂的应用来说，Web Workers内部可能需要加载外部脚本，保持代码功能的唯一性。在网页上加载多个脚本，大家都知道可以通过添加多个script标签，当页面加载时同步加载JavaScript文件，对于Web Workers来说，道理也类似，不过需要借助于importScripts方法，语法如下：

```
importScripts('other.js');
```

importScripts方法除了支持传递单个参数外，还允许接收多个脚本传递，语法如下：

```
importScripts('script1.js' , 'script2.js');
```

2.10.3 子Web Workers和内嵌Web Workers

在单个Web Workers实例内部，还允许生成多个Web Workers实例，该内部实例称为子

Web Workers。子Web Workers的出现有助于大型任务应用的拆解，不过对于子Web Workers来说，地址必须与主Web Workers同源，并且如果设置的是相对路径，会按照主Web Workers的位置进行解析，这点类似于CSS样式表中的背景图片地址。

在使用子Web Workers时还要小心，在很多浏览器中子Web Workers都会以单独的线程存在，对于生成大批量的子Web Workers会消耗过多的系统资源，因为主页面的和Web Workers之间的通信数据是复制而不是共享，每次数据传递都会伴随着序列化和反序列化过程，尤其是JSON格式的对象数据。

内嵌Web Workers可以在不下载外链脚本文件的情况下生成Web Workers。将之前Hello World示例采用内嵌Web Workers重新编写，代码如下：

```
var blob = new BlobBuilder();            // 实例化BlobBuilder对象
blob.append("onmessage =function(e) {self.postMessage(e.data);} ");
// 填充Web Workers脚本内容
var blobURL = window.URL.createObjectURL(blob.getBlob());
// 生成数据连接
var worker = new Worker(blobURL);        // 实例Web Workers对象，传递数据内容
worker.addEventListener('message', function(e) {
// 监听worker message事件，接收信息
  console.log(e.data);                   // 打印返回数据
}, false);
worker.postMessage('Hello World');       // 向Web Workers示例发送信息
```



上面示例中window的URL属性对象，目前各浏览器的实现不同，在使用时需要加上浏览器前缀，如在Chrome下使用window.webkitURL。

2.10.4 构建Web Workers应用

在现实开发中Web Workers常常被用于处理大型密集型数据任务，可以有效的避免阻塞主UI线程渲染交互。本例将演示通过Web Workers在后台处理图片数据，包含对图片进行逆色和黑白处理功能，并将处理结束后的图片数据信息发送至前台页面进行渲染。

运行示例之前，首先将代码部署在Web服务器上，如Apache、IIS或Nginx等，使用Chrome浏览器打开WebWorker.htm，效果如图2.59所示。



图2.59　Web Workers示例效果

单击“逆色”按钮，按钮下方的黑色方框区域出现上方图片的逆色效果，如图2.60所示。



图2.60 单击“逆色”按钮

单击“黑白”按钮，按钮下方显示出现图片的黑白色效果，如图2.61所示。



图2.61 单击“黑白”按钮

主页面代码如下：

```
<!DOCTYPE html>
<html>
<body>
  <img src="webworker.jpg"></img>
<!-- 示例图片 -->
  <div><button id="invert">逆色</button><button id="grayscale">黑白
</button></div>
  <canvas width=550 height=100 style="border:1px solid black"></canvas>
</body>
<script type="text/javascript">
var worker = new Worker('webworker.js');
// 实例化Web Workers对象，传递脚本
var context = document.querySelector('canvas') .getContext('2d');
// canvas容器上下文对象
var buttons = document.querySelectorAll('button');
// 所有按钮元素
var img = document.querySelector('img');                        // 图片元素
worker.addEventListener('message', function(e) {
```

```
// 监听worker message事件
        context.putImageData(e.data, 0, 0);     // 重新绘制canvas图片内容
}, false);
for(var i =0; buttons[i]; i++){
        buttons[i].addEventListener('click',function(e){
        // 监听按钮单击事件
                e.preventDefault();                  // 阻止按钮元素默认事件
                context.drawImage(img,0,0);          // 使用canvas绘制图片
                var imgData = context.getImageData(0,0,img.width,
img.height);// 获取canvas绘制的图片内容
                worker.postMessage({ id : this.id, data : imgData});
// 发送webworker图片数据
        });
};
</script>
</html>
```

主页面要完成的功能比较简单，首先，实例化WebWorkers对象并传递对应后台脚本地址，监听该对象的message事件，接收后台脚本返回的数据信息。接着，监听页面“逆色”和“黑白”按钮的单击事件，当触发单击事件时，使用canvas的drawImage方法绘制图像，并通过canvas的getImageData方法获取图像的数据信息，然后调用实例化后的WebWorkers对象的postMessage方法，向后端运行的脚本发送图片数据信息。后端WebWorker脚本代码如下：

```
var graph ={
        "invert" : function(imgdata){                       // 图形逆色处理方法
                var data = imgdata.data;
                for(var i = 0; i < data.length; i += 4) {
                        data[i] = 255 - data[i];            // 红色
                        data[i + 1] = 255 - data[i + 1];    // 绿色
                        data[i + 2] = 255 - data[i + 2];    // 蓝色
                }
                return imgdata;
        },
        "grayscale" : function(imgdata){                    // 图形黑白色处理方法
                var data = imgdata.data;
                for(var i = 0; i < data.length; i += 4) {
                        var brightness = 0.34 * data[i] + 0.5 * data[i + 1]
 + 0.16 * data[i + 2];
                        data[i] = brightness;               // 红色
                        data[i + 1] = brightness;           // 绿色
                        data[i + 2] = brightness;           // 蓝色
                }
                return imgdata;
        }
};
self.addEventListener('message', function(e) {
// 监听message事件，接收页面传递数据
        self.postMessage(graph[e.data.id](e.data.data));
// 返回接收的图形数据处理后的信息
}, false);
```

后端WebWorker脚本要做的是将主页面返回的图片数据信息交给对应的图形处理方法，在invert或grayscale图形方法处理完毕后，将返回处理后的数据给主页面，主页面在接收到返回的图片数据后，通过canvas进行渲染呈现。



Web Workers的使用并不局限于本章所列的示例，读者可以参考网站了解更多相关的使用信息和场景，网页地址http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html。



2.11 离线Web应用

传统的Web应用必须建立在联网的基础之上，HTML 5新增了一项功能，为离线Web应用的开发提供了可能性。假设用户使用在线的记事本记录信息时，网络忽然中断，对于传统的应用来说用户很可能会丢失先前书写的内容，如果使用离线Web功能开发的应用，用户可以继续离线添加笔记，待网络重新连接后将离线数据同步至线上服务。

听到这里读者一定对这个功能充满了好感和好奇，在开发一个离线应用时，开发者一般会综合使用多种功能，如离线资源缓存的文件列表Manifest文件、联网在线状态的检测、当离线状态下的本地数据存储，这几种功能缺一不可。

本章将会介绍离线Web应用相关的方法接口，同时让读者了解Manifest文件的使用，最后通过一个示例说明HTML 5开发离线应用的方法。

2.11.1 离线Web应用相关API

为了实现离线存储功能，HTML 5提供了Web存储相关的API，称为Web Storage，该功能的介绍在先前的章节中已经提过，包括LocalStorage和SessionStorage两部分，可用于对离线数据的短暂性或永久性存储。

HTML 5另外还提供了一套基于关系型的数据库Web SQL Database，可以支持页面上复杂数据的离线存储，例如可以存储用户电子邮件信息，消费账务流水信息等，同时Web SQL Database还加入了传统数据库的事物概念，使得多窗口操作可以保持数据一致性。Web SQL Database数据库是基于SQLite开发的，这点与Web Storage中的LocalStorage相同。

最后一个也是笔者认为最为强大的功能，称为IndexedDB。IndexedDB是HTML 5推出的一种轻量级的NoSQL数据库，即常说的非关系型数据库。比起传统的关系型数据库，NoSQL数据库具有易扩展、读写快速、成本低廉等特点，HTML 5的IndexedDB还同时包含了常见的数据库构造，如事物、索引、游标等，在API的使用上分为同步和异步两种形态。下面通过一个简单的示例介绍IndexedDB的使用，代码如下：

```
<!DOCTYPE html>
<html>
<body></body>
<script type="text/javascript">
var request = indexedDB.open('Html5IndexedDB', 2);   // 创建一个数据库
```

```
request.onerror = function(e) { console.log(e); };    // 监听错误事件
request.onupgradeneeded = function(event) {           // 监听事物事件
  var db = event.target.result;                       // 获取数据库对象
  var objectStore = db.createObjectStore("users", { keyPath: "html5"    });
// 创建对象存储空间存放用户信息
  objectStore.createIndex("name", "name", { unique: false });
// 创建索引来通过name搜索客户
  objectStore.createIndex("id", "id", { unique: true });
      // 创建索引来通过 email 搜索客户
  objectStore.add({ html5:'1' ,name : '小王' , sex :'女', id:'3323' ,
age:23});// 存入一条用户信息数据
};
</script>
</html>
```

将代码保存至以html为后缀的文件内，部署在Web服务器上，如Apache、IIS、Nginx等，使用最新的Chrome浏览器打开文件，然后打开Chrome浏览器的开发者工具查看缓存内容，效果如图2.62所示。

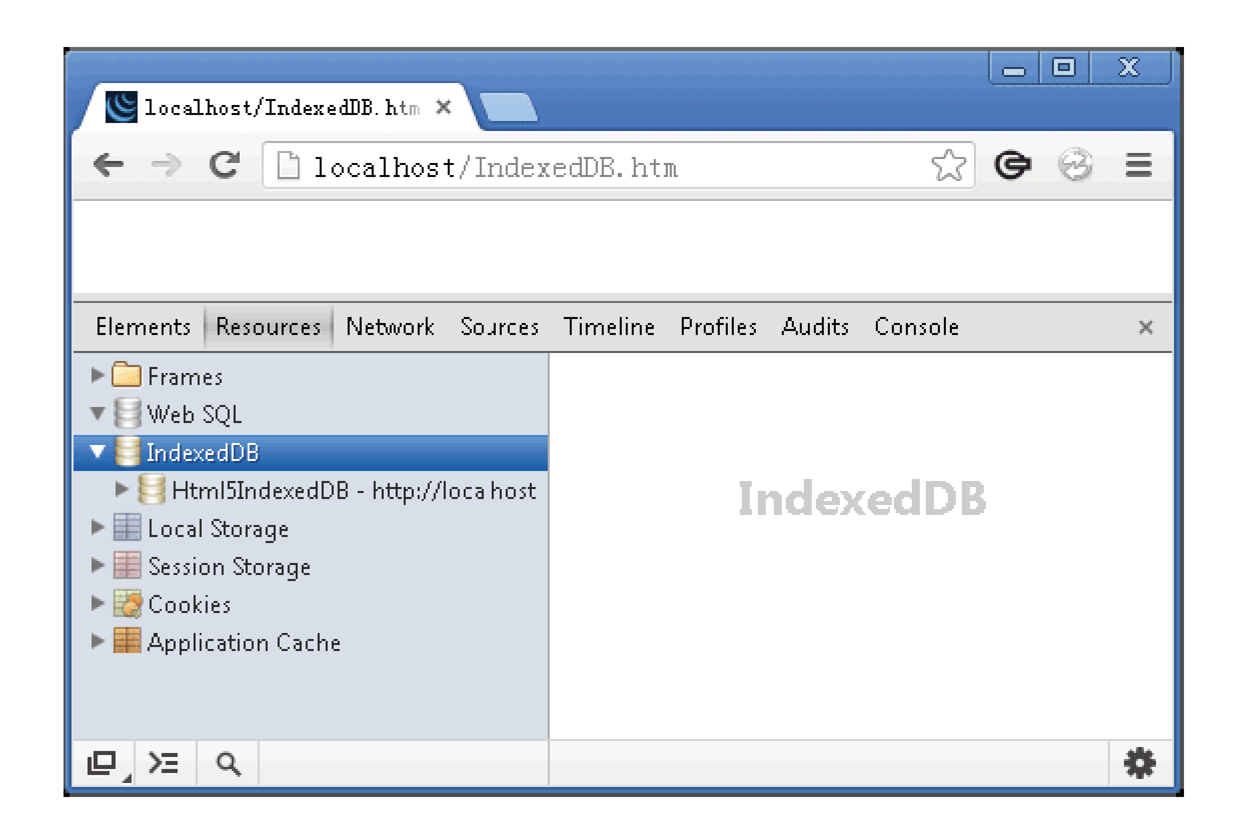


图2.62　使用最新版Chrome浏览器打开示例文件

单击左侧列表中的Html5IndexedDB项，显示数据库中的插入信息，效果如图2.63所示。

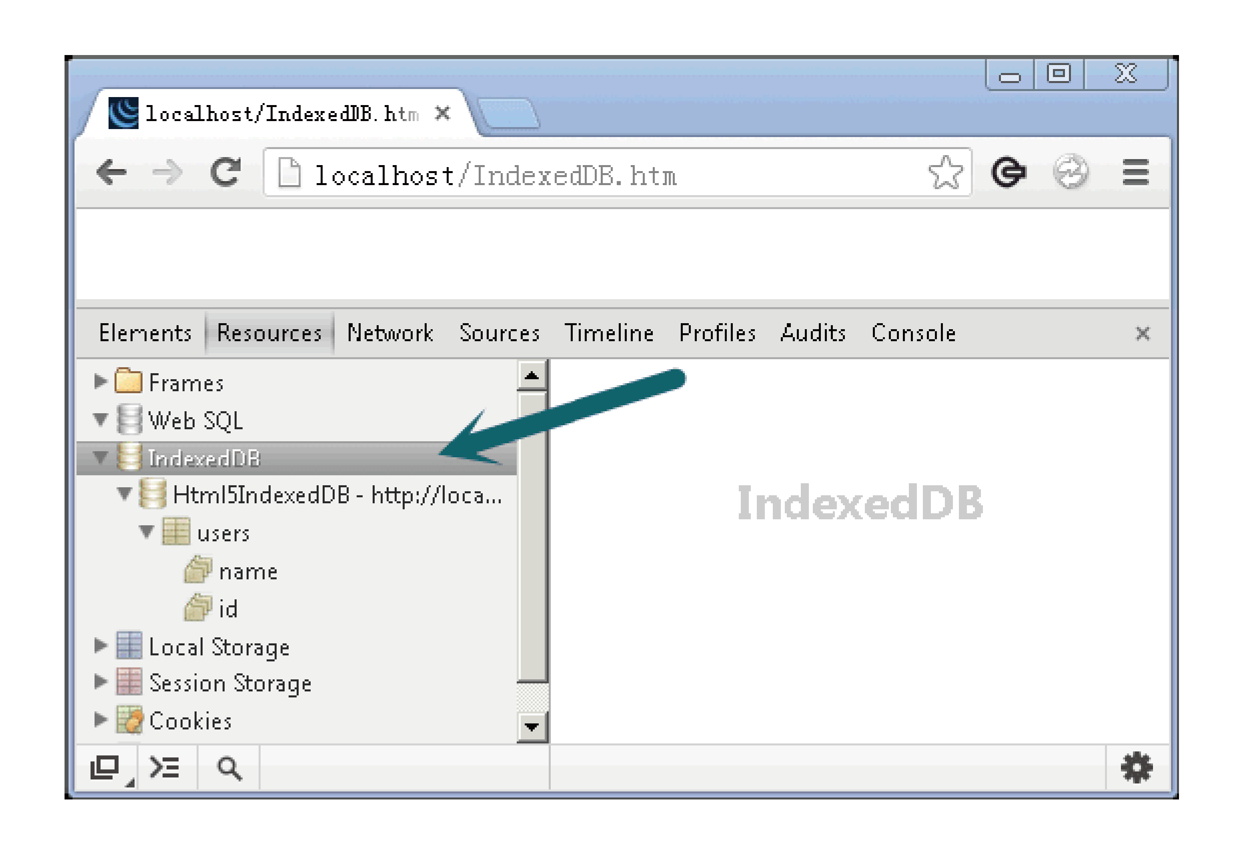


图2.63　单击左侧列表中的Html5IndexedDB

单击左侧列表中的users选项，展开数据库中users键的存储信息，同时右侧区域出现对应键值的相关存储数据，效果如图2.64所示。

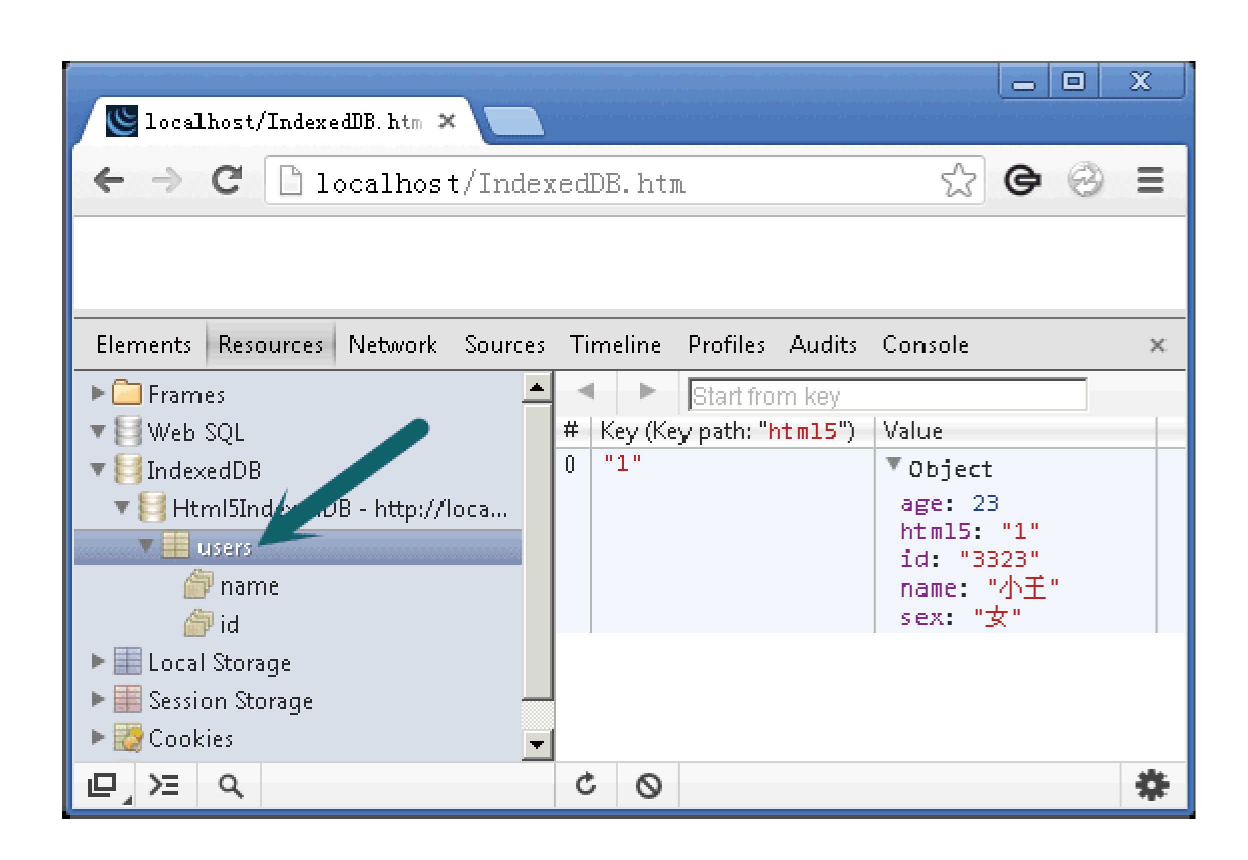


图2.64 列表中的users选项

示例代码中监听的upgradeneeded事件当每次新建数据库结构时触发，当再次打开数据库时不触发该事件，而触发另外的success事件。方法createObjectStore用于创建对象的存储空间，在示例中请求申请一个名为users的对象空间，同时传递的第一个参数保证了存储空间中每个单独的对象都是唯一的，被存储于空间中的所有对象都必须存在于html 5中。另外还可以看到使用了方法createIndex创建的数据库索引，示例中对name和id两个属性添加了索引，但在unique属性上稍有不同，id对应的unique为true，表示该键值所存储的数据具有唯一性，无法插入重复数据，而name的unique为false，表示插入的用户信息数据，并且允许用户出现名字相同的情况。

2.11.2 Manifest使用介绍

Manifest文件是HTML 5的离线缓存功能中引入的非常重要的一项，代表Web应用存储可以进行文件的离线缓存，即使在没有因特网的情况下也可以进行访问，同时可以让加载资源变得更快，已经缓存的内容不会再发生任何请求，减少了服务器的负载压力，另外，仅仅只需要从服务器上下载最新Manifest文件就能对已有资源进行更新。

Manifest文件只是一个单纯文本文件，结构非常简单，大致可分为4个部分。

- CACHE MANIFEST：MANIFEST文件顶部必须出现的标题。
- CACHE：在此标题下方出现的文件将在首次下载后进行缓存。
- NETWORK：在此标题下方出现的文件需要与服务器连接，且不会被缓存。
- FALLBACK：在此标题下方出现的文件规定当页面无法访问时的回退页面。

一个简单的Manifest文件格式如下：

```
CACHE MANIFEST
CACHE:
/demo.css
/demo.png
```

```
/demo.js
NETWORK:
/demo2.css
FALLBACK:
/ajax/     ajax.html
/html5/     /404.html
```

当文件名出现在CACHE下方后一直都会被缓存，除非发生以下情况，浏览器才会再次更新：

- 浏览器的缓存被清空，如用户手动操作的清空缓存。
- Manifest文件被修改。
- 应用程序脚本更新缓存。

换句话说，如果不发生上面出现的情况，即使开发者将服务器端的文件进行更新，用户浏览器内使用的内容也不会发生变化，如果要对应用的文件进行更新，这时候必须还要做的就是更新Manifest文件。

使用Manifest缓存功能时，有些问题需要注意：

- 如果Manifest文件中某行文件不能被下载，更新过程将失败，浏览器继续使用老缓存数据。
- Manifest文件必须与主页面同源。
- Manifest文件列表中的文件地址的相对路径，以Manifest为参照物。
- CACHE MANIFEST标题只允许出现在第一行，且必须存在。
- 使用Manifest缓存功能的页面会被认为自动进行缓存。



Manifest除了本节介绍的注意事项外，还有其特殊的加载流程，其他注意事项可以参考网站https://developer.mozilla.org/zh-CN/docs/HTML/Using_the_application_cache。

2.11.3 使用ApplicationCache API

上一节提到了使用Manifest文件对Web应用进行离线缓存，更新缓存的方法一般需要对服务器端的Manifest文件进行更新，还有一种方法就是使用浏览器提供的ApplactionCache应用接口，通过JavaScript操作ApplactionCache对象达到更新缓存的目的。ApplactionCache一共有三种方法，说明如下。

- update：发起应用缓存下载进程，尝试更新缓存。
- abort：取消正在进行的缓存更新下载。
- swapcache：更新成功后，切换为最新的缓存环境。

ApplactionCache对象上还有一个常用的属性status，该属性有6种状态，如下：

- CHECKING：检查中，状态值为2。
- DOWNLOADING：下载中，状态值为3。
- IDLE：闲置中，状态值为1。
- OBSOLETE：失效，状态值为5。
- UNCACHED：未缓存，状态值为0。

- UPDATEREADY：已更新，状态值为4。

使用Manifest功能进行的文件缓存，在每次Manifest文件更新后，第一次刷新页面用的仍旧是老的缓存内容，第二次刷新时才会更新，不过ApplicationCache为开发者提供了一种JavaScript控制的可能性。表2.24列出了目前主流浏览器对ApplicationCache的支持情况。

表2.24 主流浏览器对ApplicationCache的支持情况

浏览器	支持版本
Chrome	4.0+
Firefox	3.5+
IE	10+
Opera	10.6+
Safari	4.0+

2.11.4 搭建简单的离线应用程序

本节通过一个简单的缓存更新示例介绍Manifest和ApplicationCache的使用。在使用Manifest文件之前，首先要确保Web服务器对文件进行正确的解析，以Apache为例，需要在相应的httpd.conf配置文件中，添加如下代码配置信息：

```
AddType text/cache-Manifest .Manifest
```

采用Chrome做为演示浏览器，有的Chrome版本会默认关闭ApplicationCache功能，这时就需要进入Chrome浏览器实验室开启ApplicationCache功能，如图2.65所示。

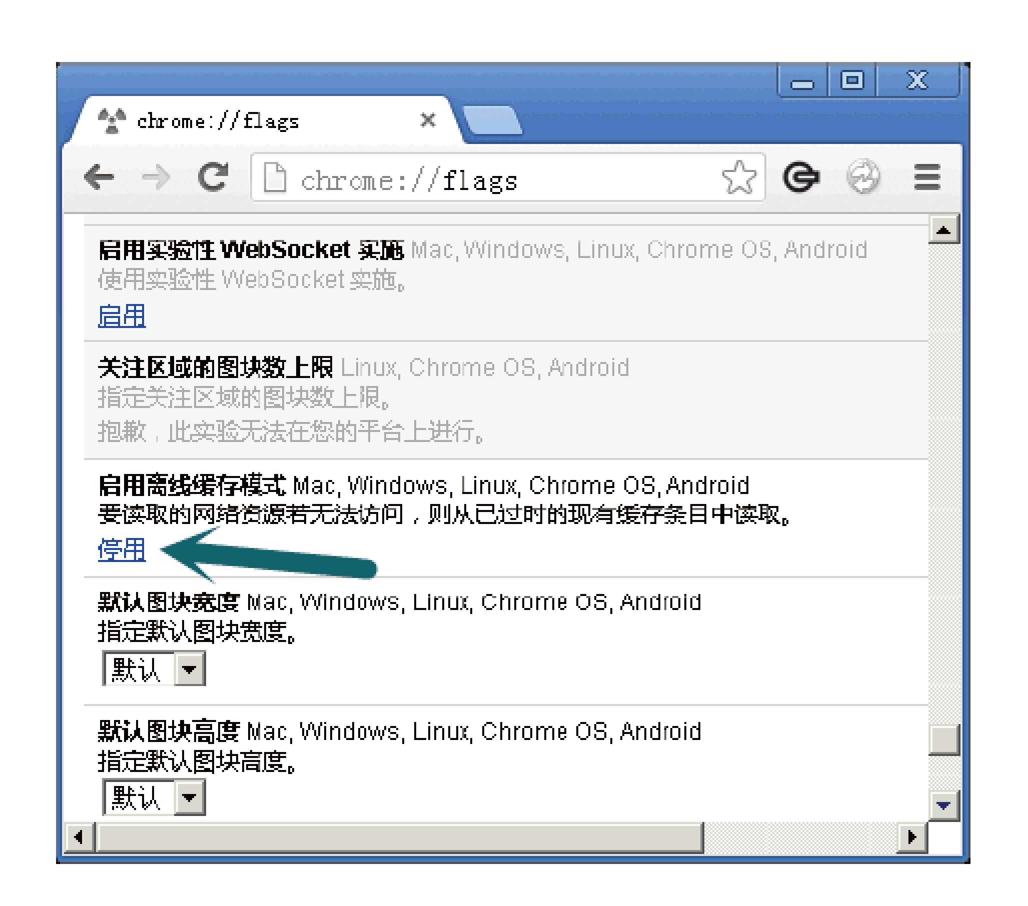


图2.65 开启Chrome浏览器的ApplicationCache功能

示例中的Manifest文件application.Manifest代码如下：

```
CACHE MANIFEST
# v1
CACHE:
demo.css
demo.jpg
demo.js
```

```
NETWORK:
demo2.css
FALLBACK:
```

示例主页面代码如下：

```
<!DOCTYPE html>
<html Manifest="ApplicationCache/application.Manifest">
      <!-- Manifest文件 ->
<head>
<script type="text/javascript" src="ApplicationCache/demo.js"></script>
<link rel="stylesheet" type="text/css" href="ApplicationCache/
demo.css">
<link rel="stylesheet" type="text/css" href="ApplicationCache/
demo2.css">
</head>
<body>
      <img src="ApplicationCache/demo.jpg"><br>
      <button>更新缓存Manifest文件</button>     <!——手动更新缓存按钮 ——>
</body>
<script type="text/javascript">
document.querySelector('button').addEventListener('click',function() {
      // 监听按钮单击事件
      var appCache = window.applicationCache;        // 获取缓存操作对象
      appCache.update();                             // 尝试更新缓存
      if (appCache.status == window.applicationCache.UPDATEREADY) {
      // 状态是否已更新
             appCache.swapCache();             // 更新成功后，切换到新的缓存
      }
})
</script>
</html>
```

将主页面文件部署在配置完毕的Apache服务器上，使用开启ApplicationCache功能的Chrome浏览器访问示例页面，同时打开开发者工具查看控制台信息，效果如图2.66所示。

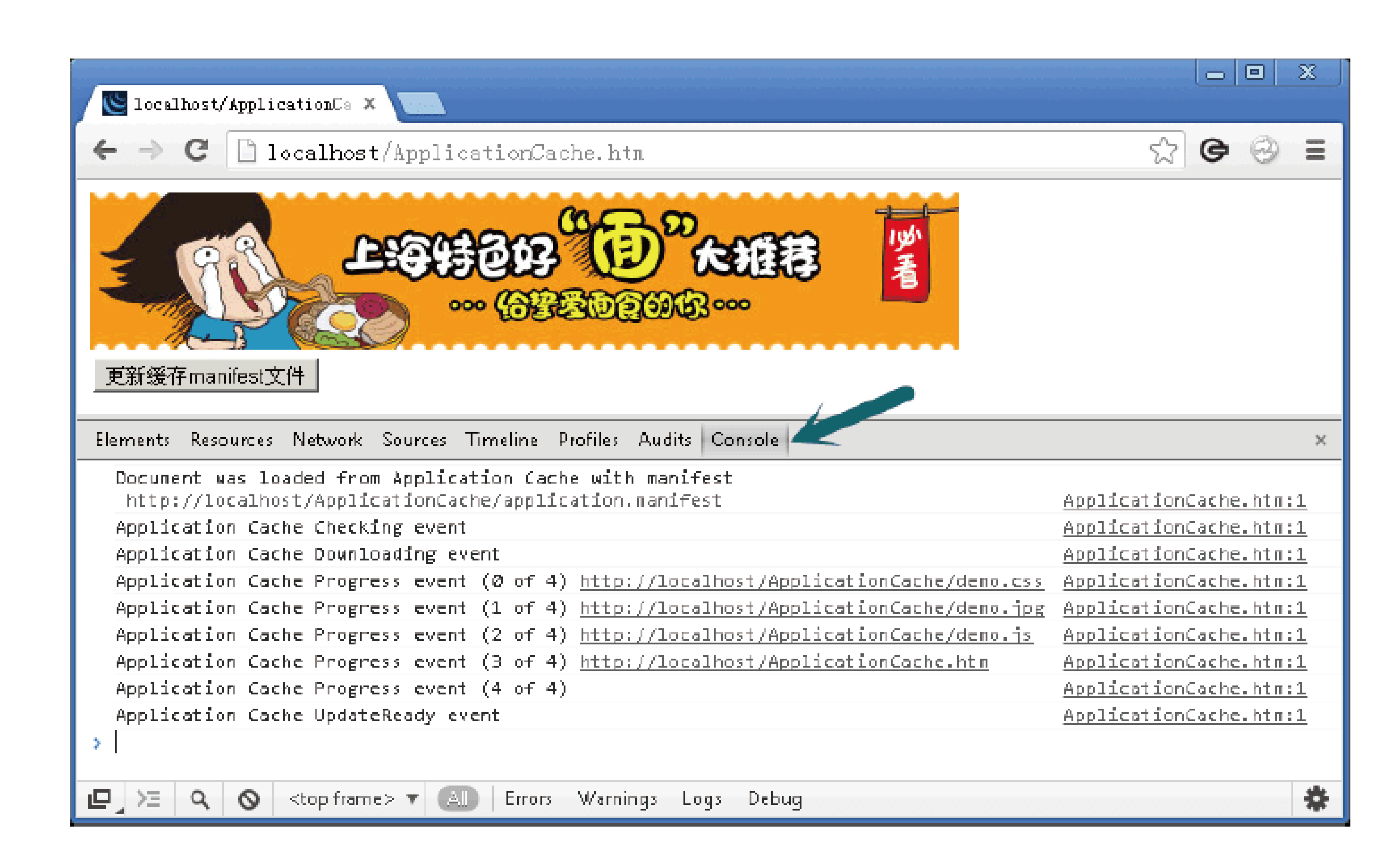


图2.66　使用Chrome浏览器打开ApplicationCache示例

在开发者工具的控制台信息中，可以看到浏览器的整个执行过程，首先下载html标签中的Manifest属性对应的Manifest文件，然后将Manifest中CACHE标签下的文件连同主页面一同进行缓存。接下来修改demo.js，添加一行代码如下：

```
alert( 'update' );
```

再更新application.Manifest文件，将第二行的v1变为v2，代码如下：

```
CACHE MANIFEST
# v2
CACHE:
demo.css
demo.jpg
demo.js
NETWORK:
demo2.css
FALLBACK:
```

单击页面“更新缓存Manifest文件”按钮，浏览器启动缓存进程更新CACHE标签下的内容，重新刷新页面，待页面加载完毕后弹出内容为update的提示框，效果如图2.67所示。

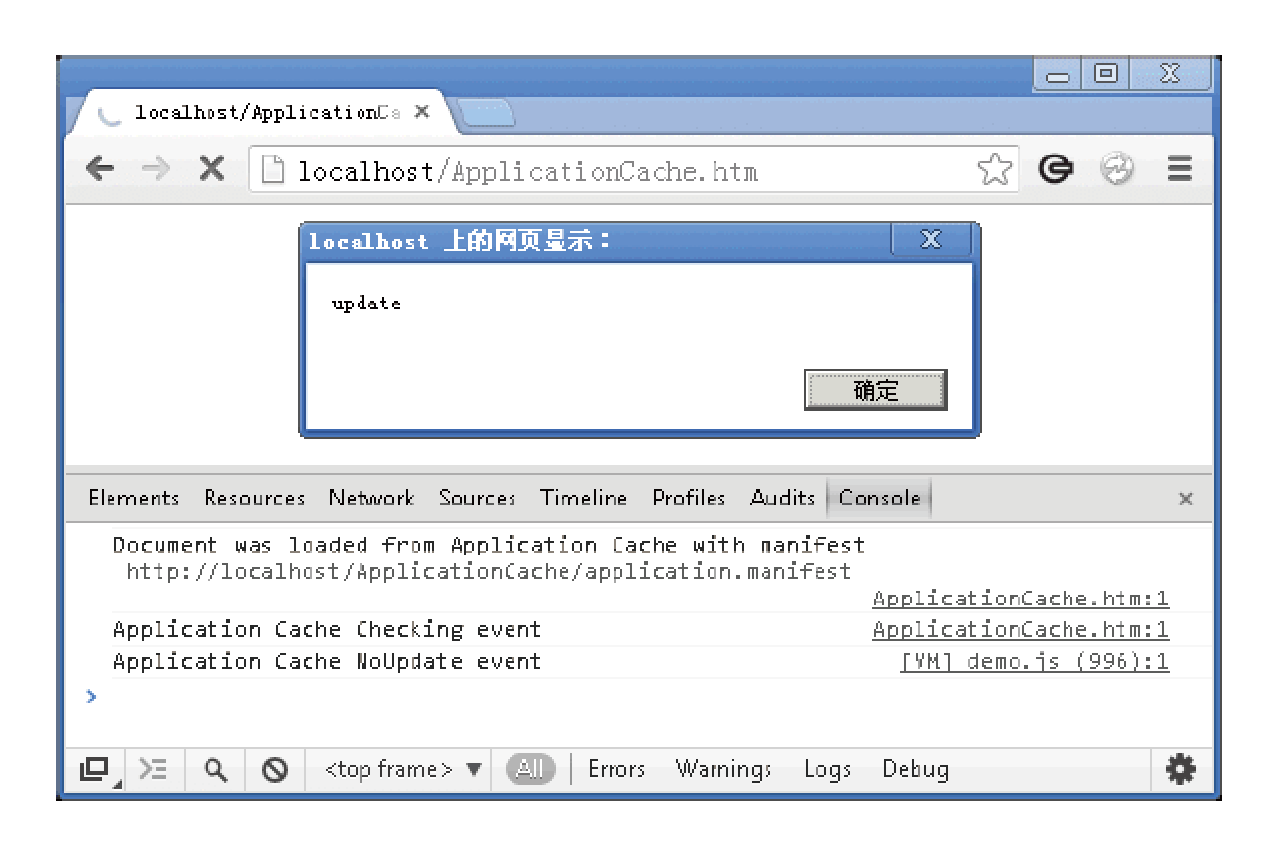


图2.67 重新刷新页面

在Chrome下可以通过访问“chrome://appcache-internals/”查看浏览器缓存的内容，本节示例的缓存信息效果如图2.68所示。

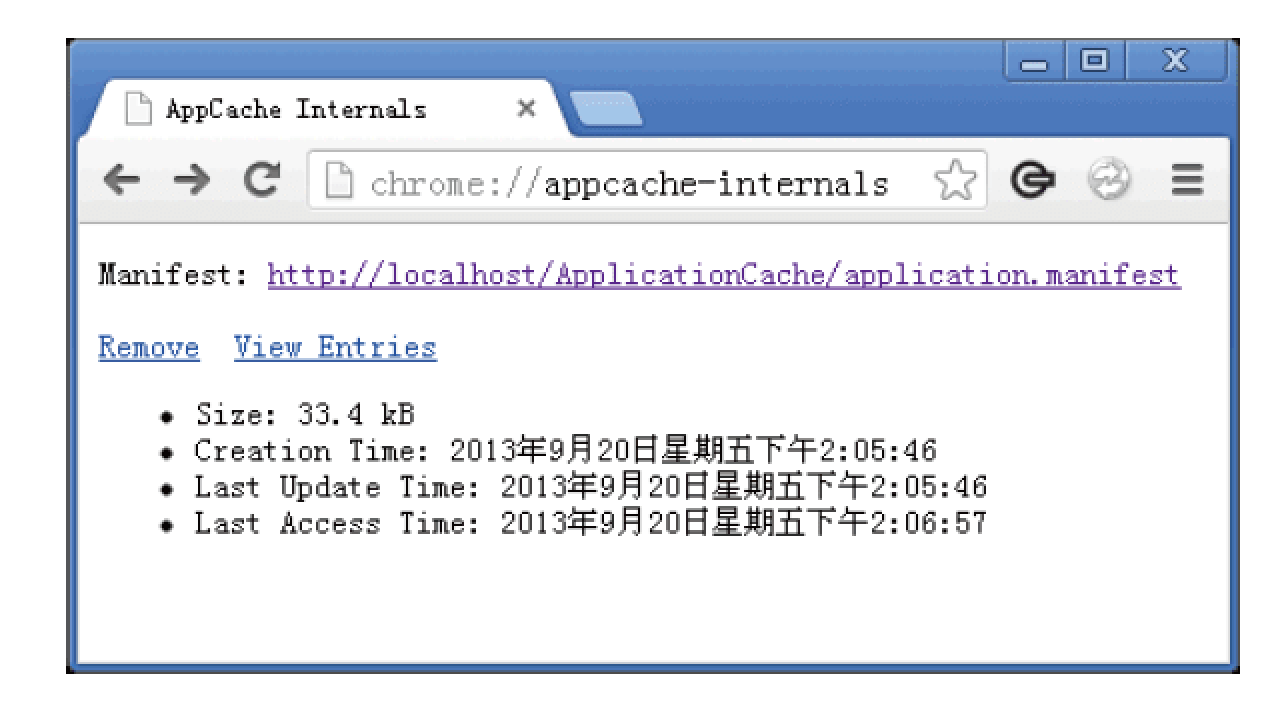


图2.68 示例缓存数据内容

单击View Entries链接按钮，展开被缓存的地址链接和缓存内容的大小，效果如图2.69所示。

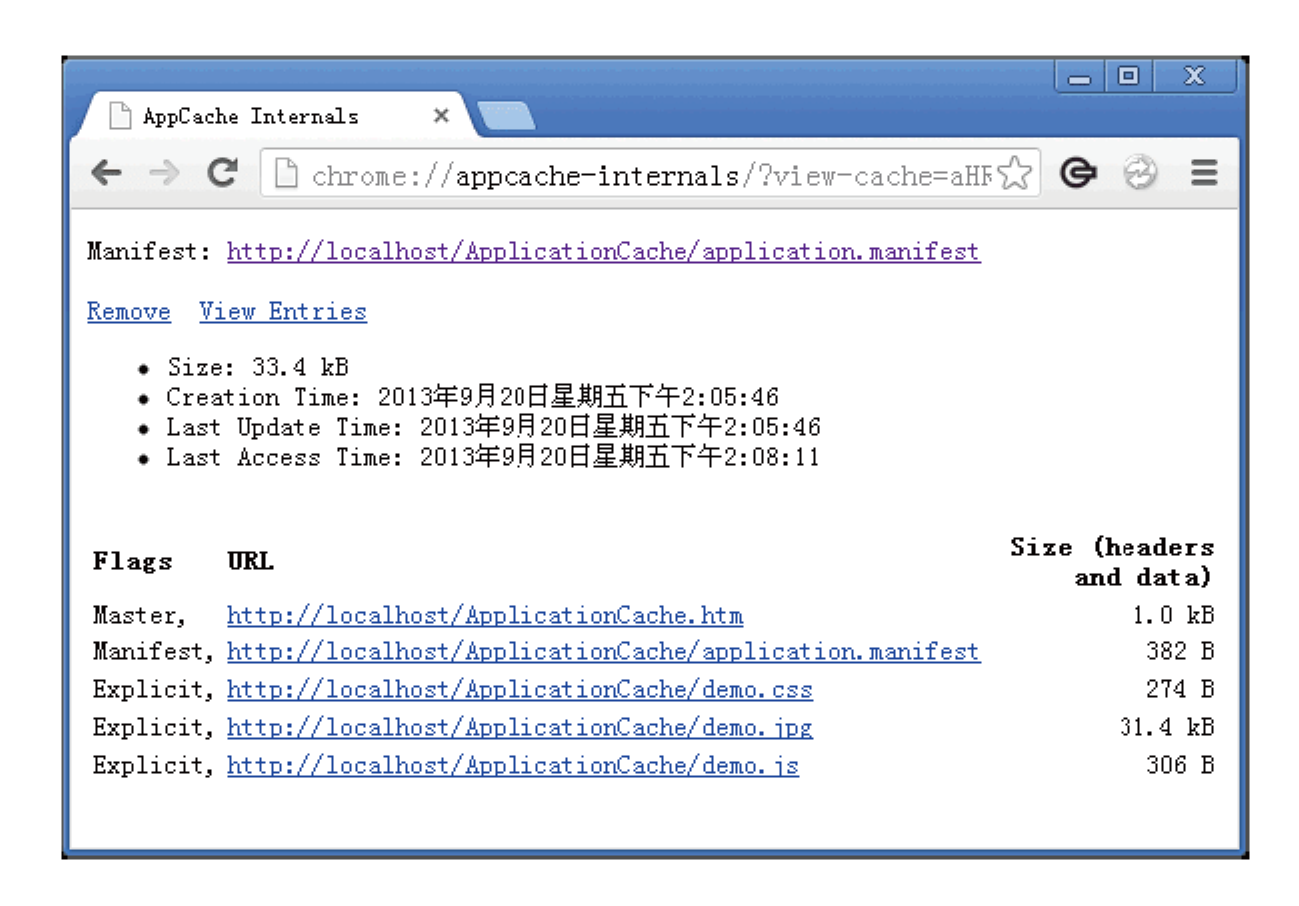


图2.69　单击View Entries链接按钮



在Firefox中可以通过访问about:cache页面（在「离线缓存设置」标题下）来检查离线缓存的当前状况。



2.12 微数据

HTML 5赋予网页更加强大的语义结构，伴随着丰富的标签和对RDFA、微格式、微数据的支持，可以构建出更加有价值的数据驱动型的Web。相较于RDFA和微格式，微数据是HTML 5推出的最新的一种语法，并且得到了搜索引擎之王Google公司的支持。微数据规范了以标签内容的方式来描述一个特定类型的信息，如个人信息、事件或评论，同时还有每个信息类型描述特定类型的项目，如事件包含场地、事件、名称等。本章将介绍语义化的相关概念和发展历史，了解微数据的相关使用方法，最后通过学习国内网站微数据的应用，加深对微数据的理解。

2.12.1 语义化概念

语义化是前端开发中的专业术语，标签语义化有助于构建良好的HTML结构，便于搜索引擎搜索和抓取，同时还有利于在不同的终端设备保持显示的一致性，搭建结构清晰的文档，便于团队开发和维护。

语义化的关键就是让网页上的数据和信息便于机器理解，提高网站的易用性，如残障人士可以借助于机器识别网页语义结构信息，从中获取网页关键内容。语义化可以让计算机对网络空间存储的数据进行智能化的分析评估，计算机可以像人脑一样理解信息的含义，完成智能筛选和处理工作。语义化形成的网络是互联网的延伸，信息在其中被赋予了更明确的含

义，让人与计算机之间能更好地协作，使得互联网变得更加智能。原本只有人类能够理解的信息，通过语义化的互联网数据计算机也能在一定程度上办到，网络从此有能力提供动态和主动的服务，如对餐饮数据的理解，计算机原本不清楚菜系的种类和口味，但语义化的意义就在于将相关的内容隐藏在页面的结构中，告诉计算机哪些文字表示菜系，哪些表示口味。

书写语义化的HTML结构其实很简单，基本的HTML各种标签就包含语义信息，如div表示一个容器、strong表示强调、ul和li表示无序列表等。在书写HTML时使用最贴近的标签来描述内容，这就是最基本的语义化。

2.12.2 Microdata的前世今生

Microdata即为微数据，Microdata这个概念最早由HTML 5规范的编辑Ian Hickson于2009年提出，但该技术的出现之前已经历了多年的发展变化。

微数据是以RDFA为基础，采用HTML形式放置RDFA的一种方式。RDFA全称为Resource Description Framework In Attributes，中文意思为属性方面资源描述框架。RDFA在2004年由Mark Birbeck在一份W3C博客中提到，后来这个概念被写入XHTML的版本中，作为W3C推荐标准。RDFA扩充了XHTML的几个属性，开发者可以利用这些属性在网页中添加机器可识别的元数据。RDFA功能非常强大，但在使用时涉及到复杂的属性交互，开发者自身都很难保证RDFA的准确性，同时，由于继承了XML的部分功能，在命名空间上容易混淆。下面通过一个例子了解RDFA的使用，首先给出没有进行RDFA语义化的数据，代码如下：

```
<div>
    我叫李三，大家叫我小三。我微博是：
    <a href="http://www.weibo.com">www. weibo.com</a>.
    我住上海市，是一名软件工程师，就职于xx科技公司。
</div>
```

将上面的结构通过RDFA进行语义优化，优化后的代码如下：

```
<div xmlns:v="http://rdf.data-vocabulary.org/#" typeof="v:Person">
    我叫<span property="v:name">李三</span>,
    大家叫我<span property="v:nickname">小三</span>。
    我微博是:
    <a href="http://www. weibo.com" rel="v:url">www. weibo.com</a>。
    我住上海市，是一名<span property="v:title">软件工程师</span>,
    目就职于<span property="v:affiliation">XX科技公司</span>。
</div>
```

Microformat称为微格式，是XHTML中另一版本的结构化数据，在2005年的超新星大会上被提出，官方地址为http://microformats.org/。微格式的出现重新将XHTML一些原有属性加以应用，如链接上的ref属性。微格式的发展受到其自身缺点的限制，没有一套统一的解析方式，繁多的解析器限制了微格式的发展。通过一个例子学习微格式的使用，原数据结构代码如下：

```
<div>
    <img src="http://i1.dpfile.com/s/i/default-avatar-s.png" />
    <strong>李三</strong>
```

```
        xx科技公司软件工程师
        上海市浦东新区xxx路
        邮编：200120
</div>
```

将上述结构通过微格式进行语义优化，代码如下：

```
<div class="vcard">
    <img class="photo" src="http://i1.dpfile.com/s/i/default-avatar-s.png" />
    <strong class="fn">李三</strong>
    <span class="org">XX科技公司</span><span class="title">软件工程师</span>
    <span class="adr">
          <span class="region">上海市</span><span class="locality">浦东新区
   </span>
          <span class="street-address">xxx路</span>
          邮编：<span class="postal-code">200120</span>
    </span>
</div>
```

Microformat之后出现的就是本章所介绍的Microdata微数据，下节将介绍微数据的相关使用。

2.12.3 如何使用Microdata优化网页

Microdata的所有信息都围绕着词汇表展开，词汇表包含了描述段落或文章的元素，但是不包含事件和组织。如果要表示事件或组织，就需要定义自己的词汇表，Microdata允许自定义词汇表，并将其使用在自己的网页中。

Microdata的使用，会用到以下属性。

- itemid：全局可识别的标识符。
- itemprop：数据项属性，可以是个单词或是个URL。
- itemref：允许微数据项通过指向特定itemid或者HTML标记。
- itemscope：声明微数据词汇表的作用域。
- itemtype：声明所使用的微数据词汇表。

下面是一段没有使用微数据优化过的HTML，代码如下：

```
<section>
    <h1>李三</h1>
    <p><img src="http://i1.dpfile.com/s/i/default-avatar-s.png" alt=
   "照片"></p>
    <p><a href="http://www. weibo.com” >微博</a></p>
</section>
```

在上述示例中添加微数据信息，需要在原有的HTML结构上添加一些属性，首先声明使用的Microdata词汇表的命名空间，通过itemtype属性完成，另外声明词汇表的作用范围，通过属性itemscope完成。本例所要展现的是一个人的个人信息，这里将使用词汇表http://data-vocabulary.org/Person来完成，该词汇表的属性如表2.25所示。

表2.25 http://data-vocabulary.org/Person词汇表属性介绍

属性	说明
name	名字
nickname	昵称
photo	头像地址
titile	职位，如工程经理
role	角色，如工程师
url	个人主页
affiliation	与该人相关的组织的名称，如雇主信息
friend	朋友
contact	联系人
acquaintance	熟人
address	住址

示例中，所有的元素都嵌套在section元素中，所以要在section上添加itemtype和itemscope，代码如下：

```
<section itemscope itemtype="http://data-vocabulary.org/Person">
   // ……省略
</section>
```

section中的第一个元素是h1，中间的数据为用户的名字，h1元素在HTML 5的Microdata数据模型中没有特殊的处理，属性值只表示相应元素的简单文本，所以在做微数据优化时要人工赋予其含义，代码如下：

```
<h1 itemprop="name">李三</h1>
```

读者可以对照表2.25理解属性name的含义。接着将使用photo属性，找到带有src属性的img标签，声明img元素的photo属性，代码如下：

```
<p><img itemprop="photo" src="http://i1.dpfile.com/s/i/default-avatar-
s.png" alt="照片"></p>
```

上述代码语法表示这里是“http://data-vocabulary.org/Person”词汇表中的一个photo属性，属性值为用户的图片地址“http://i1.dpfile.com/s/i/default-avatar-s.png”。

最后要添加的是url属性，代码如下：

```
<a itemprop="url" href="http://www. weibo.com">微博</a>
```

在表2.25中，address属性本身也是一个强大词汇表，也拥有自己的相关属性，如表2.26所示。

表2.26 Address词汇表属性

属性	说明
street-address	街道地址
locality	地方（例如，1个城市）
region	地区区域
postal-code	邮政编码
country-name	国名

2.12.4 国内网站如何使用Microdata

在国内的网站中大众点评网（http://www.dianping.com）使用Microdata的时间比较早，主要用于显示商户的评分、星级和地址等信息。

在Microdata的支持上搜索引擎Google最为出色，使用谷歌搜索信息“上海 初花”，如图2.70所示。

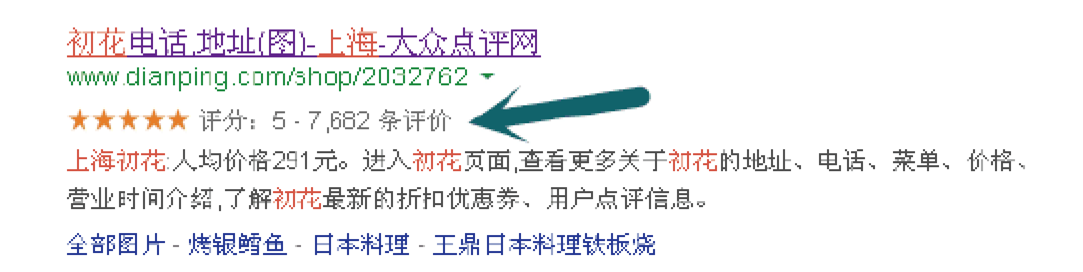


图2.70 使用谷歌搜索“上海 初花”

可以看到图中箭头所指区域的内容，比一般的搜索多出了星级和评价数，这就是微数据使用的搜索结果。通过使用谷歌的结构化数据测试工具，查看谷歌搜索引擎从网页代码中抓取了哪些信息。打开网址https://www.google.com.hk/webmasters/tools/richsnippets，效果如图2.71所示。



图2.71 谷歌的结构化数据测试工具

在页面网址文本框内，输入之前搜索的大众点评网商户网址（http://www.dianping.com/shop/2032762）。单击页面“预览”按钮，效果如图2.72所示。



图2.72 查看大众点评网商户页面微数据信息

页面的“预览”区域出现的信息比之前的更加丰富，多出了商户相关的菜系“面包屑”导航。从“提取的结构化数据”一栏中，可以看到被提取的相关词汇表，如下：

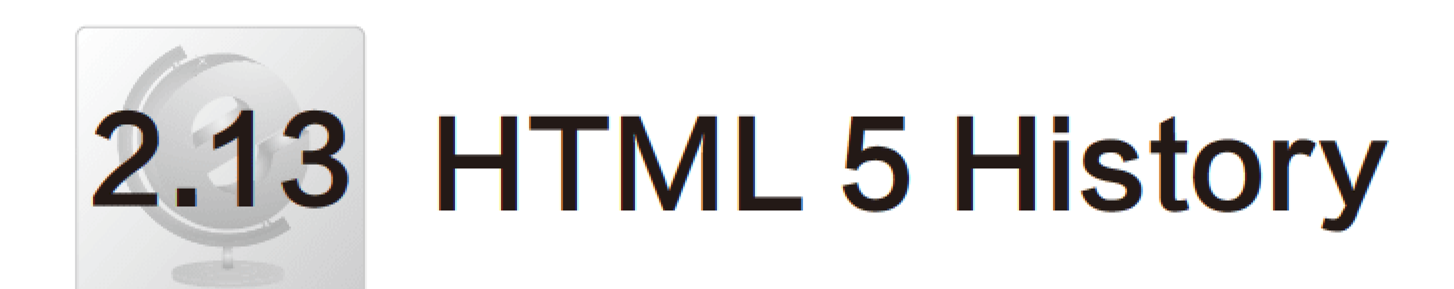
- http://data-vocabulary.org/review-aggregate：评论信息。
- http://data-vocabulary.org/breadcrumb：“面包屑”信息。
- http://data-vocabulary.org/rating：星级信息。
- http://data-vocabulary.org/address：商户地址信息。

上述词汇表所对应的大众点评网商户页面信息如图2.73所示。



图2.73 商户页面上的语义数据

读者可以查看页面对应区域源码，了解网页如何提供微数据语义功能。如果想要让自己的网站在搜索引擎上拥有更强的搜索表现，快快将微数据使用起来，武装自己的页面吧。



2.13 HTML 5 History

对于浏览器的History对象读者一定并不陌生，在HTML 5出现之前，开发者就能使用其中的一些属性和方法，所以说History并不是一个全新的东西，下面罗列了HTML 5出现之前可以使用的History相关属性和方法。

- length：历史堆栈中的记录数。
- back：返回上一页。
- forward()：前进到下一页。
- go：前进或后退到指定页，如果不写或者为0，表示刷新当前页；正数为前进，负数为后退。

通过History的相关属性和方法可以了解到，History主要提供对浏览器历史记录的访问功能，让用户在历史记录中自由的前进和后退，而在HTML 5中，这一功能得到了加强，更可

以操作历史记录中的数据。本章将介绍HTML 5新增的History方法，并介绍相关的使用，最后还会向读者介绍JavaScript中的MVC框架，了解History在其中的作用。

2.13.1 History API介绍

HTML 5给History新增加了两个方法，允许开发者逐条地添加和修改历史记录，方法如下。

- pushState：在历史堆栈的顶部添加一条记录，方法一共接收三个参数，第一个参数为对象，会在出发页面的popstate事件时，做为参数的state属性传入到事件中；第二个参数为页面的标题；第三个参数为页面的URL，不写则为当前页面。
- replaceState：更改当前页面的历史记录，参数同pushState方法。

目前主流浏览器对HTML 5 History的支持情况如表2.27所示。

表2.27 目前主流浏览器对HTML 5 History的支持情况

浏览器	支持版本
Chrome	8.0 +
Firefox	4.0+
IE	10+
Opera	11.5+
Safari	5.0+

在使用pushState方法时要注意，该方法会改变referrer的值，如果新创建的XMLHttpRequest对象在这个方法调用后使用并发送请求，HTTP请求头中的referrer的值会使用这个值，有的网页会根据referrer（来源）来校验页面的安全，这里读者一定要格外注意。

下面通过一个简单的示例了解HTML 5为History新增的两个方法的使用，页面代码如下：

```
<!DOCTYPE html>
<html>
<body>
  <img/>                                          <!--图片显示区 -->
  <div>
    <a href="/" data-type='pre'>上一张</a>           <!--上一张按钮 -->
    <a href="/" data-type='next'>下一张</a>          <!--下一张按钮 -->
  </div>
</body>
<script type="text/javascript">
var products = ['百合','枸杞','桂圆干','罗汉果','三七花','雪菊'];
// 图片数据数组
var img_path = '/History/';                       // 图片存放路径
var current_index = 0;                            // 当前图片索引
var root_href = location.href;                    // 首次页面加载地址
document.addEventListener('click',function (e) {  // 监听页面的单击事件
  var data_type = e.target.getAttribute('data-type');
// 获取触发元素的自定义属性
  switch (data_type){
    case 'pre' : e.preventDefault(); switchImg(-1); break;
// 执行上一张逻辑
```

```
19      case 'next' : e.preventDefault();switchImg(1); break;
   // 执行下一张逻辑
20    }
21  });
22  function switchImg(num) {          // 用户选择图片，并使用HTML 5 History
23    current_index += num;            // 调整当前的图片索引
24    if(current_index < 0 ){
25      current_index = products.length -1;
26    }else if(current_index >= products.length){
27      current_index = 0 ;
28    }
29    var current_img = products[current_index];
   // 获取选择的图片名称
30    history.pushState(current_img, current_img, root_href + '/'+ current_
   img); // 修改浏览器地址栏
31    document.querySelector('img').src = img_path + current_img +'.jpg';
   // 设置图片
32  };
33  window.onload = function () { switchImg(0); };
   // 加载完毕后自动切换到第一张图
34  </script>
35  </html>
```

将示例代码保存至html文件内，并部署在Web服务器，如Apache、IIS或Nginx等，使用Chrome浏览器访问示例，效果如图2.74所示。

访问示例，当页面加载完毕后，浏览器重新设置URL地址，是因为执行了第33行代码，默认将图片数组中的第一张进行展现。单击“下一张”按钮，页面内的图片发生变化，同时浏览器的URL也调整为对应图片名字，效果如图2.75所示。

用户如果要退回到上一张图片，可以单击“上一张”按钮或者单击浏览器“回退”按钮，执行的效果相同，但原理上有区别。单击“上一张”按钮是向浏览器历史堆栈中添加数据的过程，每次单击都会被记录当中，而单击浏览器“回退”按钮，是一个出栈的过程，按照后进先出的原则执行。

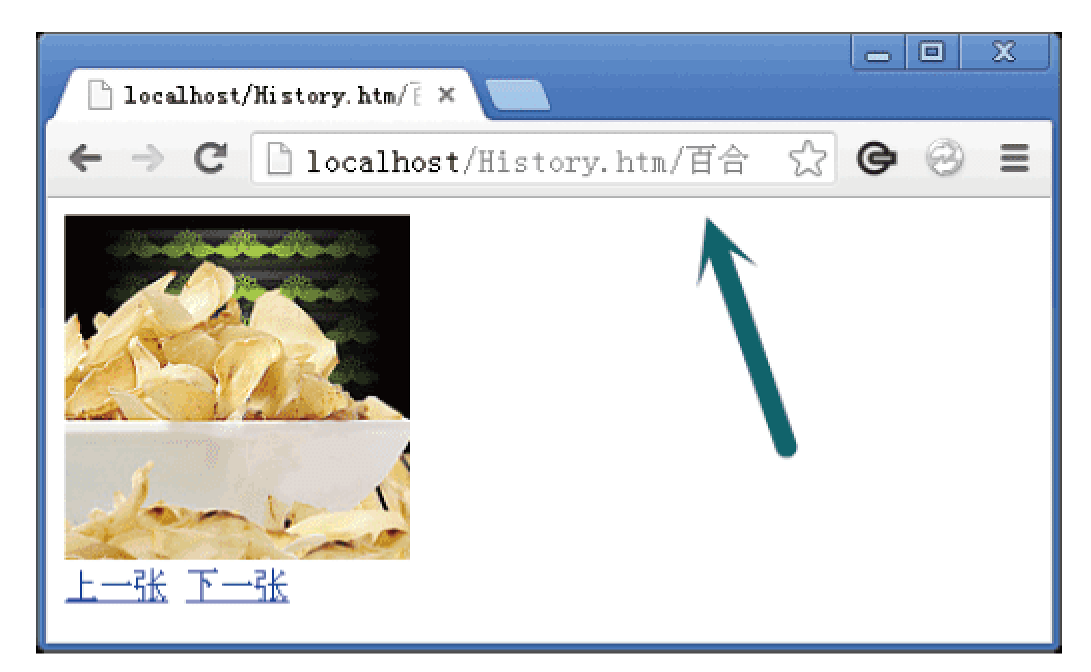




图2.74 打开History示例

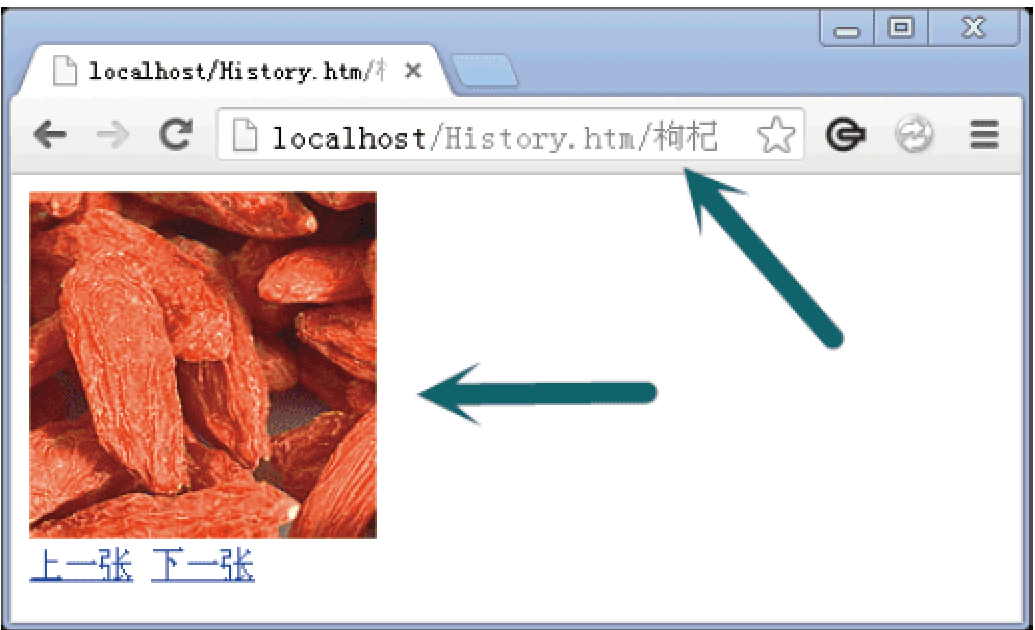


图2.75 单击“下一张”按钮

2.13.2 History与Hash

Ajax现今已被所有前端开发者所熟知，可以在不刷新当前页面的情况下更新页面上部分区域的内容。发展到现在，页面已经演变出Single Page Application应用，即单页面应用，意思是原本分散在多张页面的功能都通过一张页面来展现，常见的单页面应用如Web版QQ（地址http://web2.qq.com/），单页面应用改变了传统意识中的一个页面对应一个URL的思维，随之而来的是页面功能对应与某个URL成为问题，其核心点是如何创建和记录每次Ajax的状态。

在HTML 5 History出现之前，要记录页面状态和浏览记录只能通过Hash和IFrame两种方法，其中IFrame是IE下记录浏览状态的解决方法，下面将介绍两种方案的使用。

大家知道通常来说浏览器的记录只针对于URL而存在，其实还有一样东西也能被浏览器记录，就是Hash，即跟在井字符号“#”后面的内容。当修改页面的Hash时，不会造成页面的重新载入，比如说页面执行Ajax获取了一段数据，然后想要让浏览器记录这个过程，代码可能会像下面这样写：

```
// 执行Ajax获取数据
window.location.hash = '#data';                    // 通过hash记录这个操作
```

按照上述代码操作，浏览器可以产生一条记录，但问题还是存在，如果回到当前页面，浏览器仍然以URL地址来载入页面信息，对于浏览器来说“#”后面的数据data只用来描述某个区域，而不表示执行某个具体的业务操作，所以这里需要增加载入页面后，抓取Hash来决定具体的操作页面，代码如下：

```
window.onload = function (){                       // 监听页面资源加载完毕事件
    var hash = window.location.hash;
    if( hash == '#data'){
           // 执行对应的Ajax逻辑获取数据
    };
};
```

问题并没有就这样轻松地被解决，还有一种情况不会触发页面的load事件，当前页面的两个锚点之间进行切换时，如“demo.html#data1”跳转到“demo.html#data2”，对于这种情况，有些比较新的浏览器可以监听Hash值的变化，拥有hashchange事件，支持的浏览器有IE 8+、Firefox 3.6+、Opera 10.6+、Safari 5.0+、Chrome 5.0+。还有一些较旧的浏览器不支持该事件，只能通过不停地检查hash值判断是否发生变化，修改后的代码如下：

```
function loadData(){                               // 根据hash获取对应数据
       var hash = window.location.hash;
       if( hash == '#data'){
              // 执行对应的Ajax逻辑获取数据
       };
};
window.onload = function (){              // 监听页面资源加载“完毕”事件
       var hash = '';
       loadData();
       if ("onhashchange" in window) {    // 判断浏览器是否支持hashchange事件
              window.onhashchange = loadData;
       } else {
              setInterval(function(){                     // 轮询检查hash变化
```

```
                    if (window.location.hash == recentHash) {
// hash没有任何变化则返回
                          return;
                    };
                    hash = window.location.hash;      // 记录最新hash值
loadData();                                      // 执行Ajax方法获取数据
              }, 1000);
      }
};
```

上面的方法看上去非常完美，但遇到头疼的IE还是没辙，IE 6和IE 7版本不会将Hash的变化记录到浏览器历史中，一种URL的页面只会有一笔记录。这里就需要用到之前提到的IFrame，由于IFrame拥有自己的URL，所以IE会将其URL变化记录到浏览器的记录中。解决方案就是，在页面中动态插入一个隐藏的IFrame，同时结合之前的Hash进行浏览器的历史记录，最终修改的代码如下：

```
var hash;
window.onload = function() {
      hash = window.location.hash;
      var iframe = document.createElement('iframe');
// 创建IFrame元素
      iframe.style.display = 'none';                  // 设置样式为不显示
      iframe.src = 'javascript:void(0);';
      document.body.appendChild (iframe);             // 插入dom结构
      varn iframe_doc = iframe.contentWindow.document;
// 获取IFrame内部文档对象
      iframe_doc.open();          // 打开一个新文档，并清空当前文档
      iframe_doc.close();         // 关闭文档
      iframe_doc.location.hash = hash;                // 设置文档hash值
      loadData ();
      setInterval(function(){     // 轮询监听IFrame的hash变化
             var current_hash = iframe.location.hash;
// 获取iframe hash值
             if(current_hash != hash) {
                    location.hash = current_hash;
// 设置当前文档hash值
                    hash = current_hash;
             };
             loadData ();
      }, 1000);
};
function loadData(){                             // 根据hash获取对应数据
      var hash = window.location.hash;
      if( hash == '#data'){
             // 执行对应的Ajax逻辑获取数据
      };
};
```

使用IFrame的处理方式与单纯使用Hash的方式差异不大，只增加了一个用于存储Hash变化的隐藏IFrame元素，通过改变IFrame的src属性强行将历史记录推入IE的浏览记录中。

直到HTML 5的出现，终于有了统一的技术方案解决这一问题，如果开发的应用仍然需要兼容低版本的浏览器，可以借助于第三方的类库，如History.js，项目地址为https://github.com/balupton/History.js，该类库的API风格完全与HTML 5的History相同，读者可以通过该类库简化日常的业务开发。

2.13.3 什么是MVC

MVC全称Model View Controller。Model代表模型，View代表视图，Controller代表控制器，是软件工程中一种设计模式，最早由Trygve Reenskaug在1978年提出。MVC可以简化代码维护，增强代码的扩展性，使得代码结构变得清晰，同时一部分功能可以被重复进行利用。图2.76为MVC通用的框架结构。

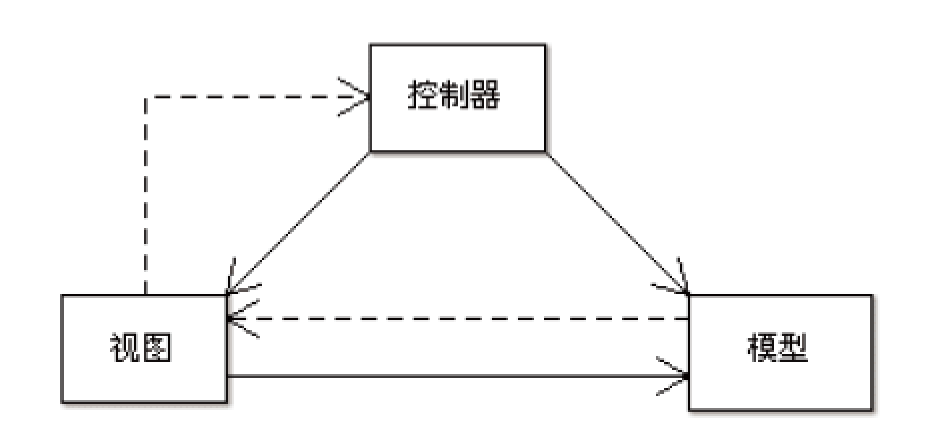


图2.76 MVC通用的框架结构

MVC的三个层次分别起着不同的作用，说明如下。

- Model（模型）：用户封装与业务相关的数据和数据的处理方法。
- View（视图）：用于处理和实现数据的显示逻辑。
- Controller（控制器）：协调模型和视图层，控制程序的流程。

MVC之所以经久不衰，是因为具备了以下优点。

- 低耦合：视图层从业务逻辑中分离，改变显示的状况。不需要调整模型层和控制器的代码，同样，如果要对底层数据进行业务变化，只需要调整模型层，如一项应用的多数据库支持，只需要给模型层添加一个适配器即可。
- 重用性高：比如时下移动互联网兴起，很多网站需要开发一套可以同时支持多种终端设备的程序，使用MVC可以让多个视图层相互独立，页面与后端的服务相互分离，模型层和控制器不需要进行任何的改变，可最大化的将代码进行重用。
- 维护成本低。
- 分工明确，开发高效：将站点的开发分成不同的工作，前端工程师负责HTML和CSS，后端Java工程师负责产品业务逻辑。

2.13.4 主流MVC框架介绍

MVC这种设计模式没有特定的形态，以往经常出现的后端开发的业务中，如使用Java、C#等。随着前端的发展，单页面应用的兴起，如Gmail，前端页面的逻辑越来越复杂，很多后端的逻辑已经前置化，这时候唯有MVC才能解决这种问题。

在JavaScript的MVC框架中，最为出名的莫过于Backbone.js，体积非常的小，压缩后只有5KB左右，官方网站为http://backbonejs.org/，Backbone.js在国内也得到了很多网站的广泛应用，如比较著名的豆瓣说。另外还有一批功能全面不过体积较大的前端MVC框架，如Knockout.js和Ember.js。Knockout.js还提出了MVC的一种变异体，叫做MVVM，即Model-View-ViewModel的缩写。最近备受关注的应该是Angular.js，网址为http://angularjs.org/，下载页面如图2.77所示。



图2.77 Angular.js下载页面

每种MVC框架都有自己的特点，声明绑定、路由等等功能已经成为MVC框架的标配，读者有兴趣可以前往站点http://todomvc.com/，里面列举了目前市面上拥有的所有JavaScript MVC框架，同时还提供了一些周边的类库供组合选择。



2.14 选择器

在HTML 5出现之前使用JavaScript查找DOM元素，原生方法有以下三种。

- getElementById：根据指定元素的id属性返回元素。
- getElementsByName：返回所有指定name属性的元素。
- getElementsByTagName：返回所有指定标签的元素。

在日常的开发中，往往需要更多复杂的查询方式，如获取某种class类名相同的元素、获取元素的指定子元素、获取带有某种自定义属性的元素。HTML 5新增了解决上述问题的JavaScript选择器，可以更加精准地定位查询元素，而不需要通过层层遍历筛选。新增的选择器支持与CSS相符的规则，方法共有两种。

- querySelector：根据选择器规则返回第一个符合要求的元素。
- querySelectorAll：根据选择器规则返回所有符合要求的元素。

新的HTML 5 JavaScript选择器接口得到了IE 8+、FireFox 3.5+、Safari 3.2+、Chrome 4.0+、Opera 10.1+浏览器的支持。

本章将介绍新增的HTML 5 JavaScript选择器，同时例举目前常用的选择器方法，最后通过查找DOM元素的实例掌握实际用法。

2.14.1 选择器分类

本节介绍的内容除了包含常用开发中的选择方式外，还会介绍部分增强选择的方式，以便读者能够全面了解选择器的使用。

1. ID选择器

使用ID选择器时，需要在前面添加“#”，同时选择器区分大小写，语法如下：

```
document.querySelector('#id');      // 等同于document. getElementById('id')
```

2. 元素选择器

元素选择器通过指定的标签查询元素，此时querySelectorAll等同于getElem1entsByTagName，语法如下：

```
document. querySelectorAll('a')     // 获取页面上所有a元素并返回列表
```

3. 样式类选择器

使用元素的样式类获取一个或一类元素，样式名字前使用“.”（英文句号）开头，语法如下：

```
document. querySelectorAll('.btn')        // 获取所有样式类中包含btn类名的元素
```

4. 分组选择器

使用querySelectorAll不仅可以获取一类相关的元素，同时还允许获取其他类别元素，两种类型之间使用逗号隔开，语法如下：

```
document. querySelectorAll('a,p');
// 获取页面上所有a元素和p元素，并通过一个列表返回
document. querySelectorAll('.btn,.txt');
// 获取页面上所有包含btn和txt样式类名的元素
```

5. 属性选择器

获取页面上包含指定属性的元素，属性名称可以是元素原生属性和用户自定义属性，语法如下：

```
document. querySelectorAll ('a[target="_blank"]')
// 获取页面上所有target属性为_blank的a元素
document. querySelectorAll ('img[data-id]')
// 获取页面上所有带有自定义属性data-id的img元素
```

6. 后代选择器

主要用于选择作为某元素后代的元素，规则左边的选择器一端包括两个或多个用空格分隔的选择器，如div a可以解释为查找所有被div包围的所有a元素，语法如下：

```
document. querySelectorAll ('div a')        // 获取页面上所有被div包含的a元素
document. querySelectorAll ('div .btn')
// 获取页面上所有被div包含的带有btn样式类名的元素
```

7. 子元素选择器

后代选择器会将元素底下的所有相关元素都搜索出来，如果使用者想进一步缩小范围，可以使用子元素选择器，只会选择某个元素的一级子元素，子元素用“>”（大于号）表示，代码如下：

```
<html>
   <div id='first'>
         <div></div>
         <div></div>
   </div>
<html>
<script>
document. querySelectorAll ('html>div')    // 只返回一个id为first的div元素
</script>
```

8. 相邻兄弟选择器

选择紧接在另一个元素后的元素，而且二者有相同的父元素，相邻兄弟选择器使用“+”（加号），该种选择器的使用相对较少，读者可以适当掌握，代码如下：

```
<div>
   <div></div>
   <div></div>
</div>
<p id="p1"></p>
<p id="p2"></p>
<script>
document. querySelectorAll ('div+p')       // 只返回一个id为p1的p元素
</script>
```

9. 伪类选择器

这里介绍其中一种伪类选择器“:first-child”表示选择元素的第一个子元素，代码如下：

```
<div>
   <p id="p1"></p>
   <p id="p2"></p>
</div>
<script>
document. querySelectorAll ('p:first-child')       // 只返回一个id为p1的p元素
</script>
```

类似的伪类选择器还有“:last-child”、“:nth-child”等。除了上述说的9种选择方法外，其实还有很多其他更高级的选择器使用方法，极大方便了日常的开发工作，使得原本在DOM中寻找元素的工作变得更加容易。



更多的选择器使用可以参考jQuery的文档，地址为http://api.jquery.com/category/selectors/。

2.14.2 使用选择器操作页面中的元素

示例代码见光盘中的querySelector.htm文件，使用Chrome浏览器打开文件，效果如图2.78所示。

选中“篮球”、“游泳”复选框，单击“获取”按钮，脚本从复选框中筛选出选中数据并输出，效果如图2.79所示。



图2.78 querySelector.htm文件效果



图2.79 选中数据并单击“获取”按钮

示例页面代码如下：

```
<!DOCTYPE html>
<html>
<body>
      <h2>兴趣爱好：<label></label></h2>         <!-- 信息输入标签 -->
      <input type="checkbox" id="c1"></input><label for="c1">篮球</label>      <!-- 复选框列表 -->
      <input type="checkbox" id="c2"></input><label for="c2">游泳</label>
      <input type="checkbox" id="c3"></input><label for="c3">唱歌</label>
      <input type="checkbox" id="c4"></input><label for="c4">桌游</label>
      <br><br>
      <button>获取</button>                    <!-- 获取按钮 -->
</body>
<script>
document.querySelector('button').addEventListener('click',function(e){  // 监听获取按钮单击事件
      e.preventDefault();                      // 组织按钮默认事件
      var checked = document.querySelectorAll('input:checked'),  // 获取所有选中复选框
             results = [];                     // 结果数组
```

```
17        checked = Array.prototype.slice.call(checked);
 // 将元素列表转为数组
18        checked.forEach(function(item){          // 循环数组获取选中数值
19              var id = item.getAttribute('id'),       // 获取复选框id
20                    label = document.querySelector('label[for="' + id
 + '"]');             // 根据id获取对应label元素
21              results.push(label.innerHTML);    // 将数值推入数组
22        });
23        document.querySelector('h2 > label').innerHTML =
 results.join(',');         // 设置显示标签内容
24 })
25 </script>
26 </html>
```

示例在第13、15、20、23行分别使用了元素选择器、伪类选择器、属性选择器和子元素选择器。



2.15 CSS 3特性

CSS随着Web 2.0的出现和发展，以往的特性和标准已经无法完全满足现今的交互和需求，开发者需要更强的字体选择、更方便的阴影渐变、更简单的图形动画。随之而来的就是CSS 3的到来，在不需要改变原有设计结构的情况下，就可以使用最新的特性，做到了良好的向后兼容。不过目前支持的浏览器还是有限的，很大一部分新功能在使用时都需要添加浏览器前缀，给开发使用带来了一定困难。本章将从开放字体格式、背景、文字效果、边框、用户界面、多列、转换和过渡8个性能点来介绍CSS 3。

2.15.1 CSS 3带来了什么

CSS 3让原有的网站更加趣味盎然，很多站点都给自己的网站页面添加了各种酷炫的CSS 3效果，让网站变得更加吸引人，如大众点评网的十周年活动页面，效果如图2.80所示。

图2.80箭头所指区域均使用了CSS 3的动画效果，用到了“@keyframes”和animation样式功能，读者可以继续向下滚动页面查看更多的CSS 3动画效果。



图2.80 大众点评网的十周年活动页面

2.15.2 开放字体格式（WOFF）

开放字体格式（WOFF），英文全称Web Open Font Format。是由Mozilla、Type Supply、LettError和其他组织协力开发的全新网络字型格式。WOFF包含了基于SFNT的字体（如PostScript、TrueType、OpenType、WOFF），字体均经过WOFF的编码工具压缩，以便嵌入网页中，并使用zlib压缩，文件一般比TTF小40%。

使用WOFF可以通过CSS 3的@font-face属性，运作方式与OpenType和TrueType字体相同，表2.28列举了目前主流浏览器的支持情况。

表2.28 主流浏览器的“@font-face”支持情况

浏览器	支持版本
Chrome	6.0 +
Firefox	3.5+
IE	9+
Opera	11.1+
Safari	5.1+

@font-face语法规则如下：

```
@font-face {
    font-family: <开放字体格式名>;
    src: <字体路径> [<字体格式>][,<字体路径> [<字体格式>]]*;
    [font-weight: <是否粗体>];
    [font-style: <字体样式>];
}
```

“@font-face”属性的使用目前仍然受到浏览器的限制，不同浏览器的支持情况各不相同，下面列举了目前常用字体的支持情况。

1. TrueType格式

该字体格式文件以ttf为后缀，是操作系统Windows和Mac上比较常见的字体格式，目前支持的浏览器有IE 9+、Firefox 3.5+、Chrome 4+、Safari 3+、Opera10+。

2. OpenType格式

该字体格式文件以otf为后缀，内置于TrueType格式基础之上，是一种原始的字体格式，提供了比TrueType格式更多的功能，目前支持的浏览器有Firefox 3.5+、Chrome 4.0+、Safari 3.1、Opera10.0+。

3. Web Open Font Format格式

该字体格式文件以woff为后缀，是一个开放的TrueType和OpenType的压缩版本，同时支持元数据包的分离，被认为是目前Web字体中的最佳格式，目前支持的浏览器有IE 9+、Firefox 3.5+、Chrome 6+、Safari 3.6+、Opera11.1+。

4. Embedded Open Type格式

该字体格式文件以eot为后缀，可以从TrueType格式中创建字体，是IE下专用字体，IE 4以上的浏览器支持。

5. SVG格式

该字体格式文件以svg为后缀，是基于SVG矢量渲染的字体格式，目前支持的浏览器有Chrome 4+、Safari 3.1+、Opera 10.0+。

为了让更多的浏览器支持“@font-face”，可以使用以下语法格式书写：

```
@font-face {
    font-family: 字体名称;
    src: url('字体文件名.eot');                    /* IE9 Compat Modes */
    src: url('字体文件名.eot?#iefix') format('embedded-opentype'),
    /* IE6-IE8 */
    url('字体文件名.woff') format('woff'),         /* 支持WOFF的浏览器 */
    url('字体文件名.ttf')  format('truetype'),/* Safari, Android, iOS */
    url('字体文件名.svg#字体文件名') format('svg');  /* Legacy iOS版本 */
}
```

下面通过一个示例介绍“@font-face”的实际使用情况，代码如下：

```
<!DOCTYPE html>
<html>
<style type="text/css">
@font-face {
    font-family: 'jackrunnerfreeregular';             /* 字体名称 */
    src: url('font-face/alterdeco-jackrunner-webfont.eot');
                 /* IE9 Compat Modes */
    src: url('font-face/alterdeco-jackrunner-webfont.eot?#iefix')
format('embedded-opentype'),      /* IE6-IE8 */
         url('font-face/alterdeco-jackrunner-webfont.woff')
format('woff'),             /* 支持WOFF的浏览器 */
         url('font-face/alterdeco-jackrunner-webfont.ttf')
```

```
format('truetype'), /* Safari, Android, iOS */
        url('font-face/alterdeco-jackrunner-webfont.
svg#jackrunnerfreeregular') format('svg');
    font-weight: 100%;
    font-style: normal;
}
body{
      font-family: 'jackrunnerfreeregular';          /* 页面字体 */
      font-size: 100px;                              /* 字体大小 */
}
</style>
<body>H T M L 5</h1></body>
</html>
```

使用Chrome浏览器打开font-face.htm示例文件，字体效果如图2.81所示。

图2.81 字体效果



http://www.fontsquirrel.com/tools/webfont-generator网站可以对字体文件进行转换，并导出各种字体格式版本，本示例的字体文件也是通过该网站处理得到。另外，想获取更多的字体可以前往站点http://www.dafont.com/。

2.15.3 背景（Backgrounds）

在CSS 2.1中background属性已经出现，并且拥有以下5个属性。

- background-color：背景色。
- background-image：背景图片地址，相对或者绝对位置。
- background-repeat：是否及如何重复背景图像，默认为repeat表示图像将在垂直方向和水平方向重复。
- background-attachment：背景图像是否固定或者随着页面的其余部分滚动，默认为scroll，表示随着页面的其余部分滚动。
- background-position：背景图像的起始位置。

在CSS 3中又给background添加了三种属性，下面分别介绍。

1. background-origin

用于设置或检索对象的背景图像计算background-position时的参考原点（位置），共有三种可选值，分别为padding-box表示从padding区域（含padding）开始显示背景图像；border-box表示从border区域（含border）开始显示背景图像；content-box表示从content区域开始显示背景图像。

通过background-origin示例比较三种属性，使用Chrome浏览器打开backgound-origin.htm文件，效果如图2.82所示。

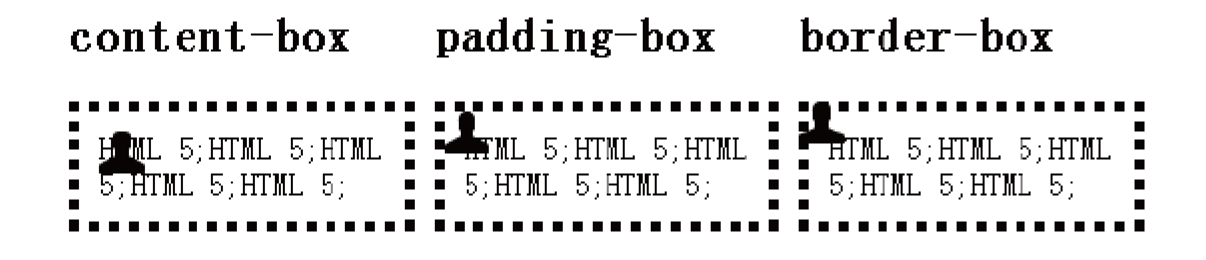


图2.82 background-origin示例效果题

2. background-clip

用于指定对象的背景图像向外裁剪的区域，同样，也具有三种可选值，分别为padding-box表示从padding区域（不含padding）开始向外裁剪背景；border-box表示从border区域（不含border）开始向外裁剪背景；content-box表示从content区域开始向外裁剪背景。

通过background-clip示例比较三种属性，使用Chrome浏览器打开backgound-clip.htm文件，效果如图2.83所示。

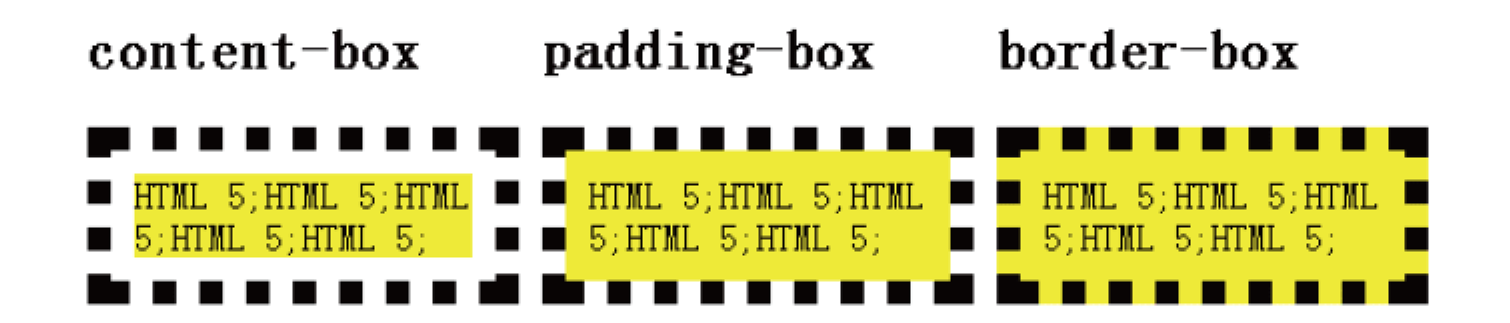


图2.83 background-clip示例效果题

3. background-size

用于检索或设置对象的背景图像的尺寸大小，允许用长度或者百分比指定背景图片大小，不允许使用负值。通过background-size示例比较各种写法，使用Chrome浏览器打开backgound-size.htm文件，效果如图2.84所示。

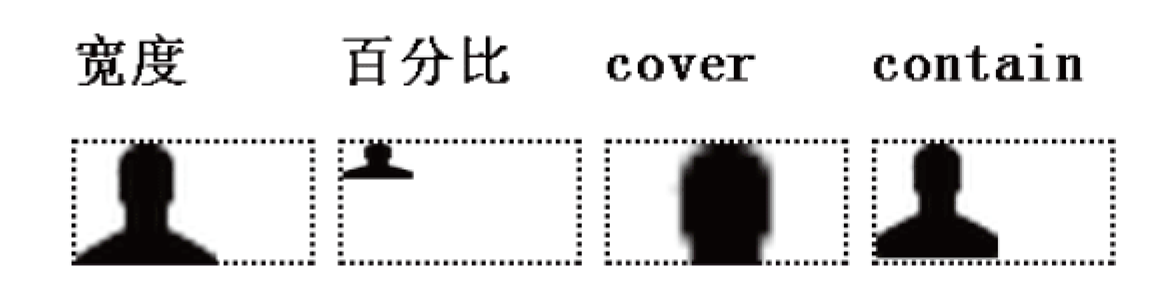


图2.84 background-size示例效果

另外，CSS 3还支持了Multiple backgrounds，即多重背景图象，可以把不同背景图象只放到一个块元素里。多个图片URL之间使用逗号隔开，如果有多个背景图片，而其他属性只有一个（例如background-repeat只有一个），表明所有背景图片应用该属性值。通过的Multiple backgrounds.htm示例学习如何使用，效果如图2.85所示。

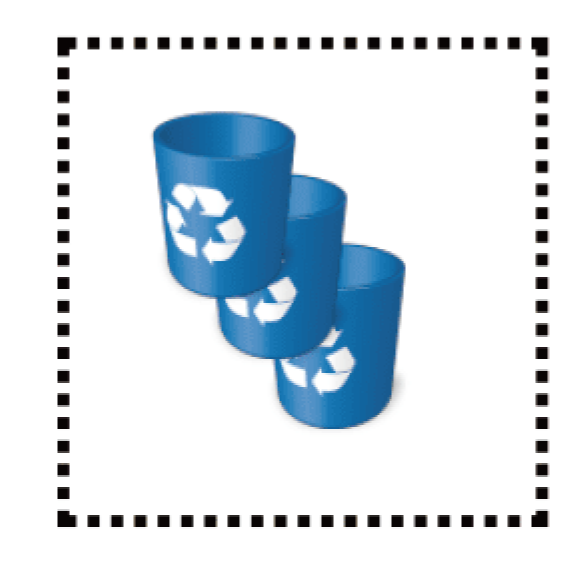


图2.85 Multiple backgrounds使用效果

2.15.4 文字效果（Text Effects）

CSS 3对文字也增加了多种效果，下面将介绍其中的4种效果。

1. text-shadow

用于设置或检索对象中文本的文字是否有阴影及模糊效果，语法如下：

```
text-shadow : none | <length> none | [<shadow>, ] * <shadow> 或none |
<color> [, <color> ]*
```

通过text-shadow示例查看使用效果，使用Chrome浏览器打开text-shadow.htm文件，效果如图2.86所示。

H T M L 5

图2.86 text-shadow示例效果

2. text-overflow

用于设置或检索是否使用一个省略标记“...”表示对象内文本的溢出，语法如下：

```
text-overflow :  clip | ellipsis
```

- clip：不显示省略标记“...”，而是简单的裁切。
- ellipsis：当对象内文本溢出时显示省略标记“...”。

通过text-overflow示例查看使用效果，使用Chrome浏览器打开text-overflow.htm文件，效果如图2.87所示。

text-overflow : clip

不显示省略标记，而是简单的

text-overflow : ellipsis

当对象内文本溢出时显示…

图2.87 text-overflow示例效果

3．word-wrap

用于检索或设置对象中单词之间的间隔，语法如下：

```
word-wrap : normal | break-word
```

- normal：控制连续文本换行。
- break-word：内容将在边界内换行。

4．word-break

用于断行的规则，语法如下：

```
word-break: normal|break-all|keep-all;
```

- normal：使用浏览器默认的换行规则。
- break-all：允许在单词内换行。
- keep-all：只能在半角空格或连字符处换行。

2.15.5 边框（Border）

CSS中的Border主要用于处理边框效果，经常被使用在网页中，新版的CSS 3对Border属性添加了更多丰富的功能，本节将介绍以下几种功能属性。

1．border-colors

用于设置或检索对象边框的多重颜色，CSS 2中border-colors已经出现，但在CSS 3中可以制作渐变边框，不过目前只有Firefox对border-colors的支持比较完整。查看border-colors示例，使用Firefox浏览器打开border-colors.htm文件，效果如图2.88所示。

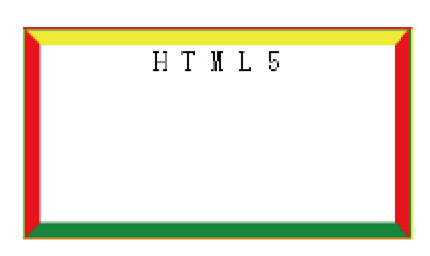


图2.88 border-colors示例效果

2．border-radius

用于设置或检索对象使用圆角边框，这个属性应该是目前出镜率最多的CSS 3属性，也

是目前各大网站最常用的CSS 3属性，语法如下：

```
border-radius :  none | <length>{1,4} [ / <length>{1,4} ]?
```

查看border-radius示例，使用Chrome浏览器打开border-radius.htm文件，效果如图2.89所示。

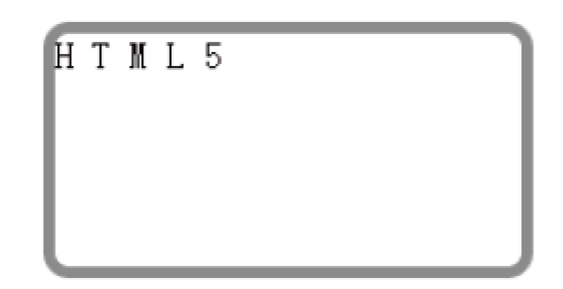


图2.89 border-radius示例效果

3．border-image

用于设置或检索对象的边框样式使用图像来填充，一直以来边框的填充只能使用颜色来进行，CSS 3在这点上做了较大的突破，border-image可以被拆解为另外5种属性。

- border-image-source：用于设置引入图片的地址。
- border-image-slice：用于切割引入的图片。
- border-image-width：设置边框的宽度。
- border-image-repeat：设置图片的排列方式，如stretch、repeat、round。
- border-image-outset：设置边框图像超过边框盒的偏移量。

border-image属性语法如下：

```
border-image:border-image-source border-image-slice{1,4}/border-image-
width{1,4} border-image-repeat{0,2}
```

查看border-image示例，使用Chrome浏览器打开border-image.htm文件，效果如图2.90所示。



图2.90 border-image示例效果

4．box-shadow

用于向边框添加一个或多个阴影，通过逗号分隔阴影列表，语法如下：

```
box-shadow: h-shadow v-shadow blur spread color inset;
```

各属性值说明如下。

- h-shadow：水平阴影的位置，允许负值。

- v-shadow：垂直阴影的位置，允许负值。
- blur：模糊距离，可选。
- spread：阴影的尺寸，可选。
- color：阴影的颜色，可选。
- inset：将外部阴影（outset）改为内部阴影，可选。

查看box-shadow示例，使用Chrome浏览器打开box-shadow.htm文件，效果如图2.91所示。

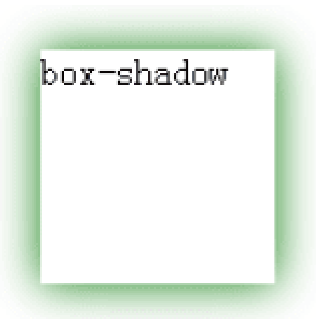


图2.91 box-shadow示例效果

2.15.6 用户界面（User interface）

1. outline

用于设置或检索对象外的线条轮廓，语法如下：

```
outline: [outline-color] || [outline-style] || [outline-width] ||
[outline-offset] | inherit
```

各属性值说明如下。

- outline-style：指定轮廓边框轮廓。
- outline-width：指定轮廓边框宽度。
- outline-offset：指定轮廓边框偏移位置的数值。
- outline-color：指定轮廓边框颜色。

查看outline示例，使用Chrome浏览器打开outline.htm文件，效果如图2.92所示。

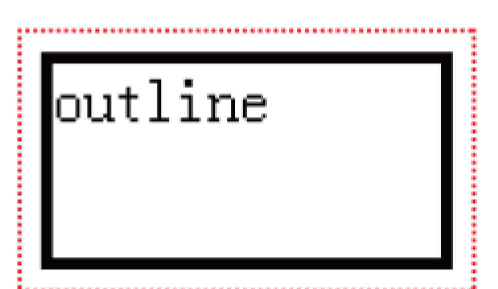


图2.92 outline示例效果

2. box-sizing

用于改变容器的盒模型组成方式，语法如下：

```
box-sizing: content-box | border-box
```

各属性值说明如下。

- content-box：padding和border不被包含在定义的width和height之内，与标准模式下的盒模型相同。
- border-box：padding和border被包含在定义的width和height之内，表现为怪异模式下的盒模型。

用于设置或检索对象的盒模型组成模式，通过两段样式比较两种方式的区别，比较如下：

```
.content-box { box-sizing:content-box; width:200px; padding:10px;
border:15px solid #eee; }
```

content-box使用效果说明如图2.93所示。

```
.border-box { box-sizing:border-box; width:200px; padding:10px;
 border:15px solid #eee; }
```

border-box x使用效果说明图2.94所示。

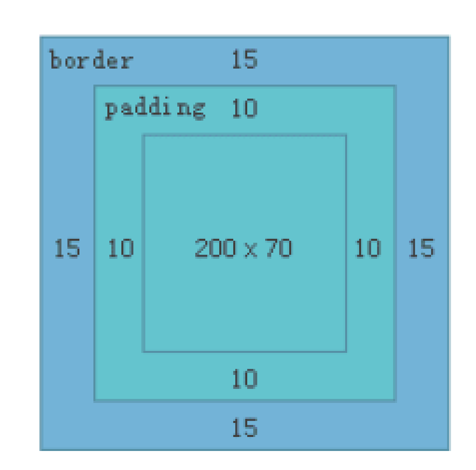


图2.93 content-box图示

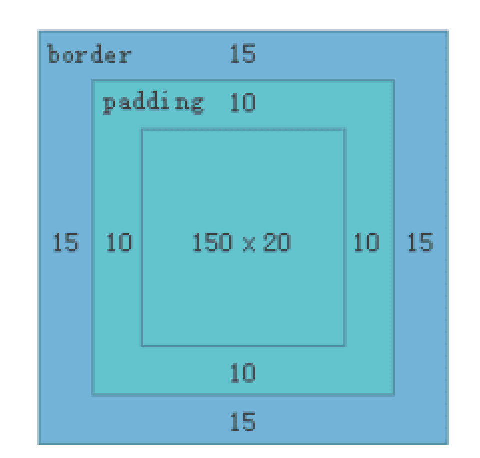


图2.94 border-box图示

3. resize

用于设置或检索对象的区域是否允许用户缩放，调节元素尺寸大小，语法如下：

```
resize : none | both | horizontal | vertical | inherit
```

各属性值说明如下。

- none：不允许用户调整元素大小。
- both：用户可以调节元素的宽度和高度。
- horizontal：用户可以调节元素的宽度。
- vertical：用户可以调节元素的高度。

使用效果如图2.95箭头所示区域。



图2.95 resize使用效果

4. nav系列

用于设置对象的导航顺序和方向，分为nav-up、nav-right、nav-down、nav-up。作为旧版本HTML属性tabindex的替代品，允许在CSS中设置页面元素和通过键盘操作获取焦点的顺序，通常使用Tab键进行顺序切换。

5. ime-mode

用于设置或检索是否允许用户激活输入中文、韩文、日文等的输入法（IME）状态，语法如下：

```
zoom: auto | normal | active | inactive | disabled
```

各属性说明如下。

- auto：默认值，不影响当前输入法编辑器的状态。
- normal：输入法编辑器的状态应该是normal，这个值可以用于用户样式表来覆盖页面的设置。
- active：输入法编辑器的状态初始时是激活的，输入将一直使用该输入法直到用户切换输入法。
- inactive：输入法编辑器的状态初始时是非激活状态，除非用户激活输入法。
- disabled：禁用输入法编辑器，该输入法编辑器也许不会被用户激活。



经过笔者测试，目前新版的Chrome、Opera、Safari均不支持该属性。

2.15.7 多列（Multiple Columns）

用于文本的多列布局，是CSS 3增加的一个多列布局模块，排版布局是CSS中非常重要的功能，在报刊、杂志的布局中尤为重要，多列提供了以下几类功能。

- 列数和列宽：column-count、column-width。
- 列的间距和分列样式：column-gap、column-rule-color、column-rule-style、column-rule-width、column-rule。
- 列的分栏符：break-before、break-after、break-inside。
- 跨越列：column-span。
- 填充列：column-fill。

查看Multiple Columns示例，使用Chrome浏览器打开Multiple Columns.htm文件，效果如图2.96所示。

雾凇俗称树挂，是可遇不可求的自然奇观。雾凇非冰非雪，而是由于雾中无数零摄氏度以下而尚未结冰的雾滴随风在树枝等物体上不断积聚冻粘的结果，表现为白色不透明的粒状结构沉积物。雾凇形成需要气温很低，而且水汽又很充分，同时能具备这两个形成雾凇的极重要而又相互矛盾的自然条件更是难得。美丽的雾凇难得，但我国东北地区的吉林市因其特殊的地理环境和人为条件却造就了相当于“可遇可求”的中国四大自然奇观之一的“吉林雾凇”。

图2.96 Multiple Columns示例效果

2.15.8 转换（Transform）

CSS 3 Transform包含旋转（rotate）、扭曲（skew）、缩放（scale）、移动（translate）和矩阵变形（matrix），语法如下：

```
transform: rotate | scale | skew | translate |matrix;
```

当使用多个transform属性时，需要使用空格符号隔开，各属性说明如下。

- rotate：通过设置角度参数给元素指定一个2D旋转，正值为顺时针，负值为逆时针。
- skew：通过传入的矢量进行水平方向和垂直方扭曲变形，即X轴和Y轴同时按一定的角度值进行扭曲变形，同时还支持使用skewX和skewY进行单个方向的扭曲变形。
- scale：通过传入的矢量进行水平方向和垂直方向缩放，同时还支持使用scaleX和scaleY进行单个方向的缩放。
- translate：通过传入的矢量进行水平方向和垂直方向移动，同时还支持使用translateX和translateY进行单个方向的移动。
- matrix：以一个含6位值的变换矩阵的形式指定一个2D变换，该属性涉及到数学中的矩阵变化，可以说该方法是transform转换属性的根基，一切的Transform使用都是由matrix变化而来的。

在使用transform属性前，还有一个非常关键的属性transform-origin，用于设置每次转化前的基点位置，默认基点位置为中心位置，参数可以使用百分值、px或者方向值（如left、right、top、center和bottom）

查看Transform示例，使用Chrome浏览器打开Transform.htm文件，效果如图2.97所示。

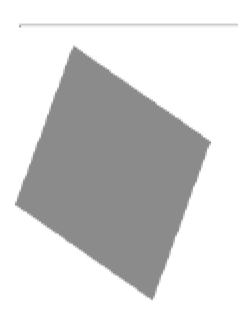


图2.97　Transform示例效果

2.15.9 过渡（Transition）

用于将CSS的属性值在一定的时间区域内平滑的过渡，制作各种动态的效果。比如，单击鼠标移入元素，使得元素颜色发生渐变，语法如下：

```
transition : [<'transition-property'> || <'transition-duration'> ||
<'transition-timing-function'> || <'transition-delay'> [, [<'transition-
property'> || <'transition-duration'> || <'transition-timing-function'> ||
<'transition-delay'>]]*
```

各属性说明如下。

- transition-property：指定当前元素某个属性改变时执行的过渡效果。
- transition-duration：指定转化过程的持续时间，单位为s。

- transition-timing-function：根据时间的推进改变属性值的变换速率，在有6种，分别为ease（逐渐变化）、linear（匀速）、ease-in（加速）、ease-out（减速）、ease-in-out（先加速后减速）、cubic-bezier（自定义贝塞尔曲线值）。
- transition-delay：设置延后执行时间，即当元素属性值发生改变后多少时间开始执行，单位为s。

查看Transition示例，使用Chrome浏览器打开Transition.htm文件，效果如图2.98所示。

将鼠标移入方框区域内，图中的两个圆形各自的位置、颜色、形状发生变化，变化后的效果如图2.99所示。

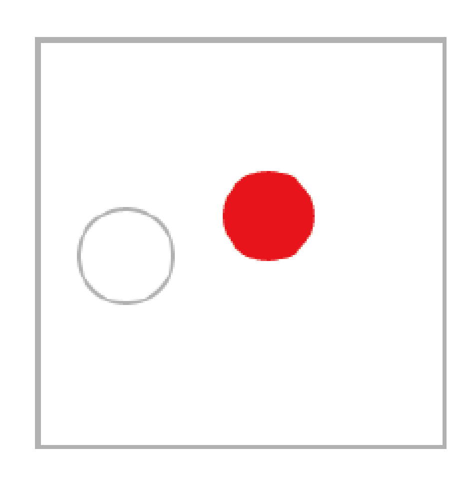


图2.98 Transition示例效果

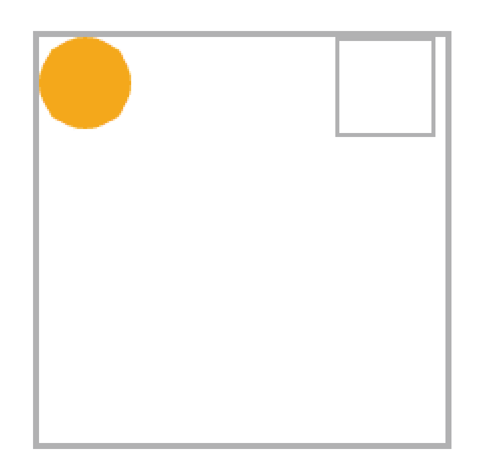


图2.99 Transition示例变化后

2.16 本章小结

CSS 3出来已经有一段时间了，很多的网站开始大规模地使用CSS 3对网页进行重构。本章从开放字体格式、背景、文字效果、边框、用户界面、多列、转换和过渡这8个方面介绍和学习CSS 3带来的新功能，借助这些功能，开发者可以将原来使用JavaScript实现的动画效果，甚至有一些无法实现的效果，简单地通过CSS 3进行开发实现，极大地缩短了开发时间和降低了维护成本。虽然，目前并非所有的浏览器都能使用CSS 3的效果，但网站开发者可以遵循渐进增强的原则进行网站优化。



第3章 HTML 5相关概念和框架

HTML 5的概念风声水起，各种概念与框架应运而生。HTML 5本身就是一个很宽泛的词，从严格意义上说，HTML 5代表的是最新版本的Web语义标准，目前最具权威和影响力的是1994年成立的万维网联盟World Wide Web Consortium（W3C）制定的标准。到目前为止，W3C已经发布了200多项Web技术标准及实施指南。HTML 5发展至今已有数十年，各种围绕标准而产生的框架也异常之多，本章将从几个大方向介绍典型且应用广泛的概念与框架，帮助读者了解在实际项目中如何选择和使用这些框架。

本章知识点：

- *响应式Web设计*
- *移动JavaScript框架*
- *CSS 3 UI框架*
- *HTML 5图表库*



3.1 响应式Web设计

近几年来，设备与分辨率的更新过快，在不同的分辨率下，用户体验问题越来越困扰用户和开发者。如果为了获得良好的用户体验，传统的互联网开发模式只能针对各种设备和分辨率进行定制，响应式Web设计是对这种问题的变革。响应式设计可以说是2013年比较流行的一种设计趋势，越来越多的个人和公司也加入了响应式设计的阵列。那么，到底什么是响应式设计，响应式设计给我们带来了什么，响应式设计真的是万能的吗？本章将为你一一揭晓这些答案。

3.1.1 什么是响应式Web设计

响应式Web设计（Responsive Web Design）是指一个页面的设计与开发可以根据用户行为（如改变浏览器窗口大小）和不同设备环境（如系统平台、屏幕分辨率以及现在流行的屏幕定向等）做出相应的响应，适应用户可感知的流畅的阅读和操作体验。响应式设计一般需要考虑几种状态，以适应不同的分辨率不同的设备和用户行为。因此需要设计、前端和开发之间有序的协调，简单来说，它不是一个部门的事情，而是多个部门共同协作的结果。所以响应式设计一般都需要遵循一个有序可寻的流程，一般响应式设计遵循“设计先行，内容优

先，移动优先”的原则。在移动终端屏幕，交互设计师先根据终端进行交互设计，把需求上最重要的内容展示在移动小屏幕上。之所以移动优先也是这个原因，小屏幕可以让交互设计师提炼出什么是最重要的内容，可以让交互设计师更专注于设计。前端和开发根据不同的设备、分辨率和应用场景设计移动框架和响应式框架。

一般响应式Web设计包括以下几个模式。

- 布局（Layout）
- 导航（Navigation）
- 图片（Image）
- 多媒体（Media）
- 表单（Form）
- 模块/组件（Modules）



有关响应式模式的更多资料请参考网站http://bradfrost.github.io/this-is-responsive/patterns.html。

3.1.2 流式布局

响应式设计最主要的两个技术分别是流式布局和媒体查询。这里所指的流式布局是利用一套灵活的流体网格系统搭建网站的布局。流体网格系统的网格一般采用相对单位长度（如百分比或em）进行设计。这种网格系统的好处是可以根据视窗的不同宽度按比例进行动态的调整网格的宽度。所以不管宽度如何改变，每一个网格的比例都是固定的，可以很好地适应不同的设备。流式布局一般不使用绝对单位（如像素）进行设计，因为使用绝对单位有很多的局限性，而且网格的宽度不会随着设备的不同而改变。

那么如何将平常所使用的固定网格系统转化成流式布局呢，Ethan Marcotte提出了一个非常方便的计算公式，即，用目标元素的宽度除以其父级元素的宽度，公式如下：

```
相对宽度 = 目标元素的宽度 ÷ 其父级元素的宽度，
```

比如HTML代码如下：

```
<div class=” container” >
<aside>…</aside>
<section>…</section>
</div>
```

固定布局样式代码如下：

```
.container{
    width: 900px;
}
aside{
    width: 240px;
    float: left;
    margin: 10px;
}
```

```
Section{
    width: 660px;
    float: right;
}
```

转化成流式布局代码如下：

```
.container{
    max-width: 900px;
}
aside{
    width: 26.6667%;          /* 240 ÷ 900 */
    float: left;
    margin: 1.1111%;          /* 10 ÷ 900 */
}
Section{
    width: 73.3333%;          /* 660 ÷ 900 */
    float: right;
}
```



Ethan Marcotte是《RESPONSIVE Web DESIGN》的作者，他最早提出响应式设计相关概念。有关作者的详细信息请参考http://alistapart.com/author/emarcotte，相关的书籍的详细信息请参考http://www.abookapart.com/products/responsive-web-design/。

3.1.3 媒体查询

响应式设计的另一个重要技术手段是媒体查询。如果只是简单地设计一个流式布局系统，那么可以保证每个网格按比例地放大和缩小，但有可能会使得在小屏幕下（如手机设备）因网格太小而严重影响阅读，这样的设计称不上响应式设计。媒体查询可以来解决这一问题。媒体查询可以为特定的浏览器和设备提供特定的样式。媒体查询是CSS 3的一个新特性，是对媒体类型的扩展。



W3C列出了10种媒体类型，请参考http://www.w3.org/TR/CSS2/media.html#media-types。

在响应式设计中，媒体查询一般在CSS中定义，如最常见的使用方式设置屏幕宽度小于1024px时的样式，代码如下：

```
@media screen and (max-width: 1024px){
    // 在这里设置小于1024px时的样式
}
```

设置屏幕宽度小于600px时的样式，代码如下：

```
@media screen and (max-width: 600px){
    // 在这里设置小于600px时的样式
}
```

设置屏幕宽度在600px~900px之间时的样式，代码如下：

```
@media screen and (max-width: 600px) and (min-width: 900px){
    // 设置样式
}
```

设置设备的实际分辨率小于480px时的样式（如iPhone），代码如下：

```
@media screen and (max-device-width: 480px){
    // iPhone手机样式在这里设置
}
```

设置iPad或iPhone在横向时的样式，代码如下：

```
@media screen and (orientation:landscape){
    // 在这里设置样式
}
```



更多关于媒体查询请参考http://www.w3.org/TR/css3-mediaqueries/。

3.1.4 Twitter Bootstrap理念

Bootstrap是一款提供给开发人员快速搭建页面的前端框架，它利用HTML、CSS和JavaScript来帮助构建用户界面组件。开发人员可以使用Bootstrap作为基础，快速创建站点或者Web应用程序。最早的Bootstrap框架始于2011年8年，当时Twitter开源了Bootstap。

Twitter Bootstrap包括了图表、栅格系统、排版、表格、表单、按钮、导航、图像、响应式设计和其他一些交互组件。最新版本的Twitter Bootstrap 3已经很好地支持了响应式设计，同时也遵守移动优先和图片响应的设计原则。



更多关于Twitter Bootstrap地址的信息请参考其官网地址http://getbootstrap.com/。

3.1.5 Twitter Bootstrap应用

Twitter Bootstrap的应用非常广泛，本小节主要介绍如何使用Bootstrap。

1. 下载Bootstrap

Bootstrap的下载方式有多种。

（1）官方网站下载地址为http://getbootstrap.com，如图3.1所示。

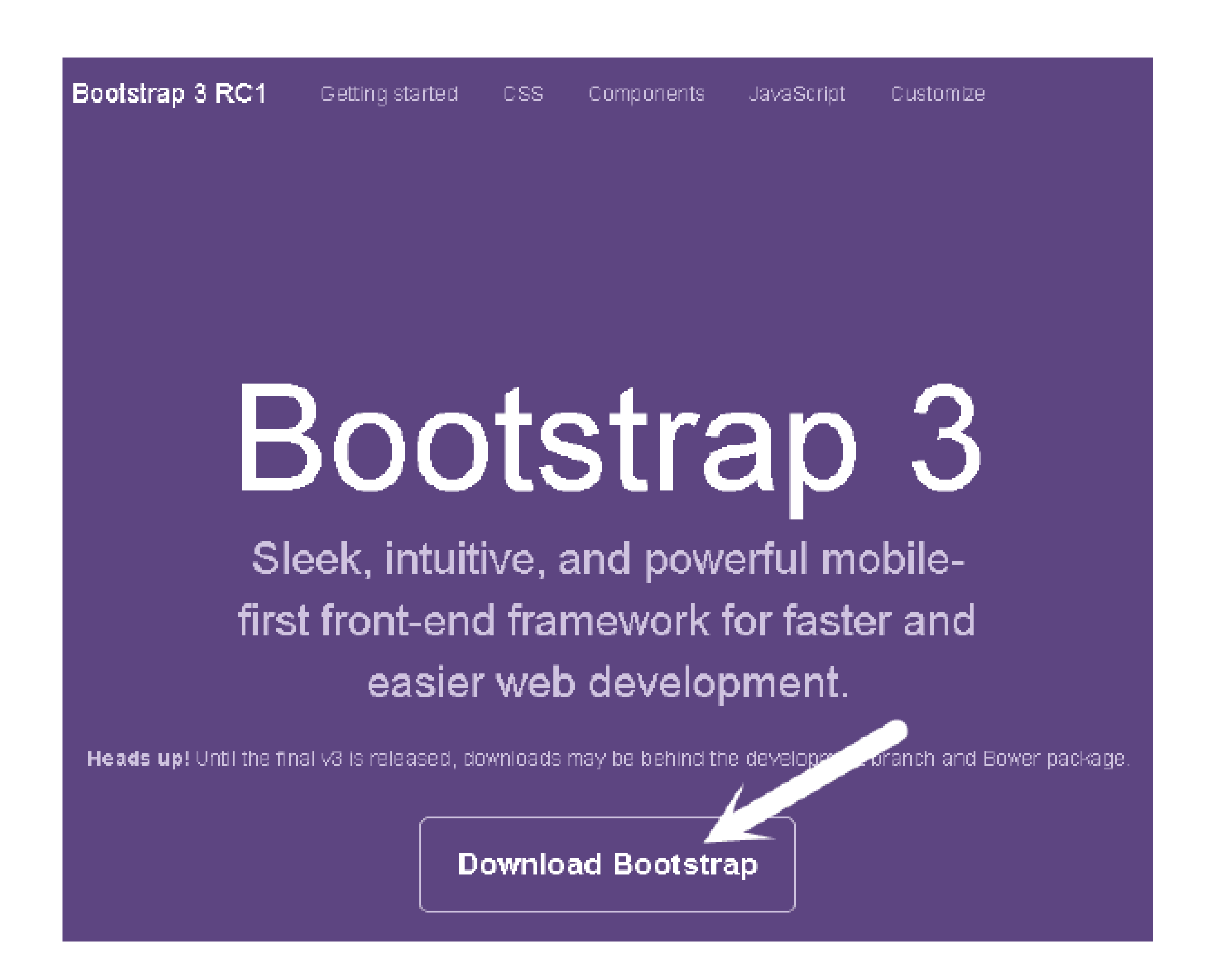


图3.1 Bootstrap官方下载地址

（2）如果想获取最新开发中的Bootstrap版本，可以通过GitHub网站https://github.com/twbs/bootstrap，如图3.2所示。

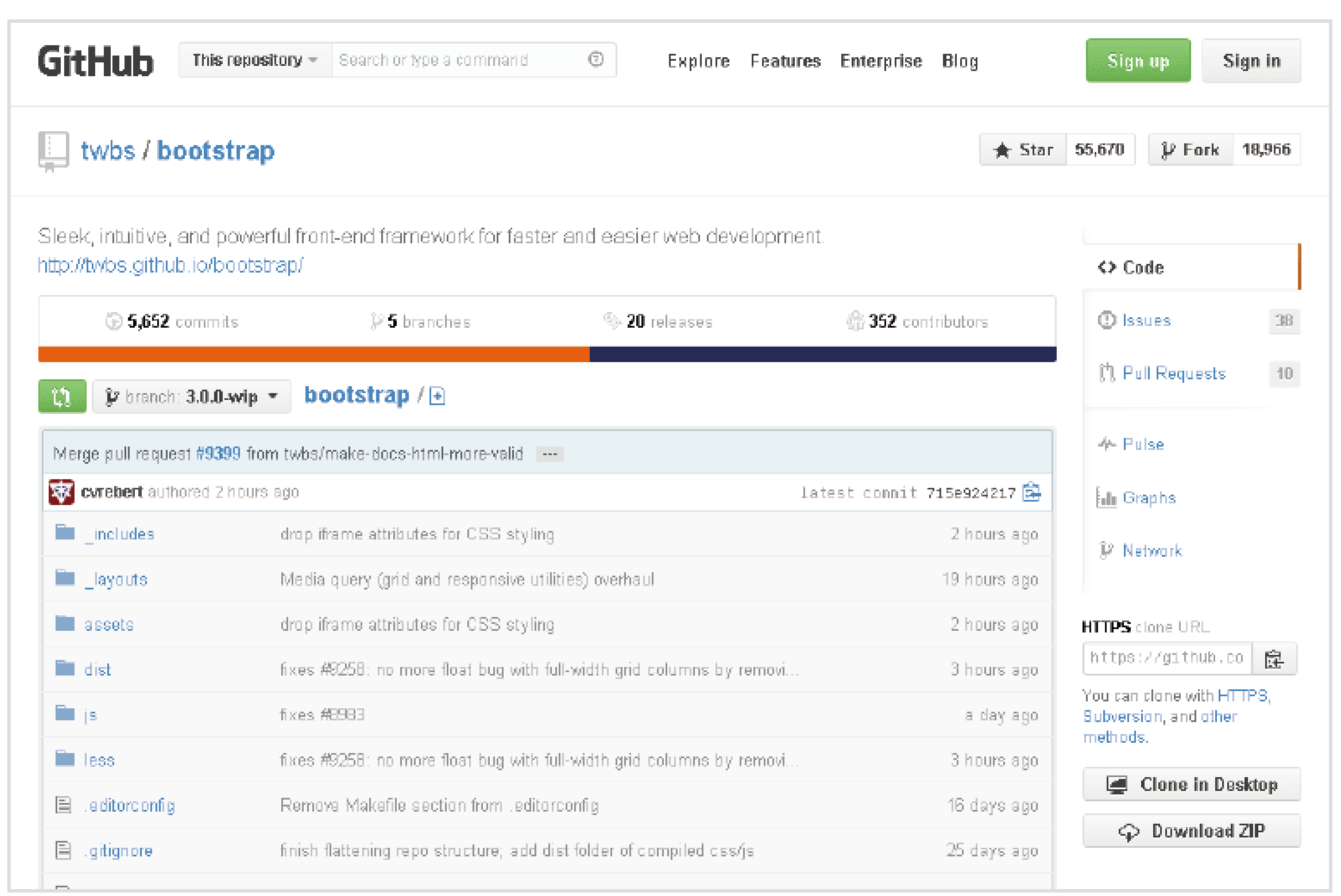


图3.2 GitHub中的Bootstrap

（3）通过Bower安装，Bower（http://bower.io/）是Twitter公司开发的包管理工具，命令如下：

```
$bower install bootstrap
```

（4）直接使用CDN文件，代码如下：

```
<!-- 最新版本的css文件 -->
<link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0.0-
   rc1/css/bootstrap.min.css">
<!--最新版本的JavaScript文件 -->
```

```
<script src="//netdna.bootstrapcdn.com/bootstrap/3.0.0-rc1/js/bootstrap.
min.js"></script>
```



CDN全称Content Delivery Network，中文意思内容分发网络。基本的设计思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输得更快、更稳定。通过在网络各处放置节点服务器所构成的在现有的互联网基础之上的一层智能虚拟网络，CDN系统能够实时地根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息，将用户的请求重新导向离用户最近的服务节点上。其目的是使用户可就近取得所需内容，解决Internet网络拥挤的状况，提高用户访问网站的响应速度。

2. 在HTML文件中引入Bootstrap文件

在HTML中引入Bootstrap的CSS、JavaScript文件，代码如下：

```
<!DOCTYPE html>
<html>
  <head>
    <title>使用Bootstrap 模板</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="css/bootstrap.min.css" rel="stylesheet" media="screen">
   <!--引入Bootstrap的CSS文件 -->
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="js/bootstrap.min.js"></script> <!--如果需要JavaScript，
   则引入Bootstrap的JavaScript组件-->
    <script src="js/respond.js"></script><!--如果是设计一个响应式网站，且需要
   支持IE 6/IE 7/IE 8，则引入respond.js-->
  </body>
</html>
```

Bootstrap支持响应式设计，如果设计的站点或者Web应用程序，需要支持IE 6/7/8等浏览器，但因其不支持Media Query，则需要引入Respond.js库。



Respond.js是提供一套快速且轻量级的针对不支持Media Queries浏览器的优雅降级的解决方案，其GitHub地址为https://github.com/scottjehl/Respond。

3. 响应式设计布局举例

比如要制作一个流式嵌套布局，和固定栅格的嵌套有些不同，被嵌套的列数之和为12，流式嵌套布局使用百分比来设置每列的宽度，代码如下：

```
<div class="row-fluid">
  <div class="span12">
    流12
    <div class="row-fluid">
      <div class="span6">
        流6
        <div class="row-fluid">
```

```
            <div class="span6">流 6</div>
            <div class="span6">流6</div>
          </div>
        </div>
        <div class="span6">流6</div>
      </div>
    </div>
</div>
```

浏览器运行效果如图3.3所示。

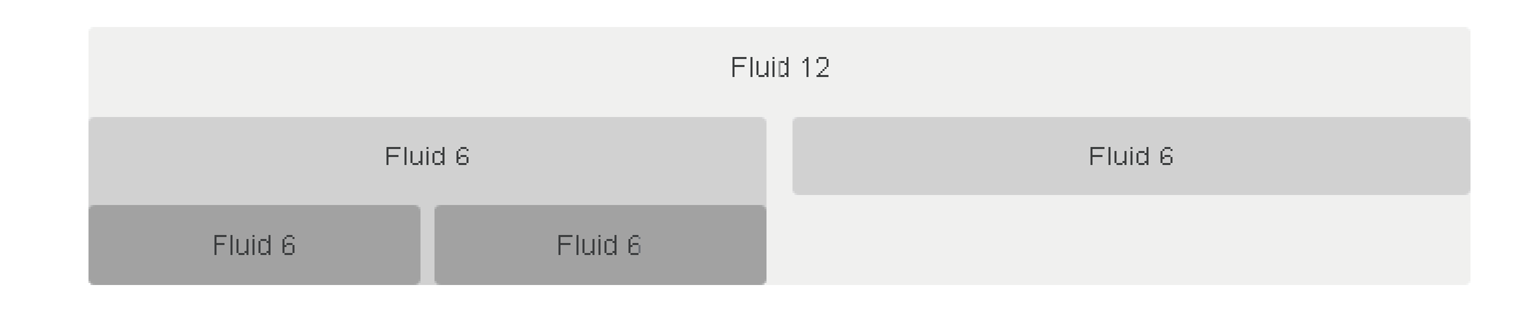


图3.3 Bootstrap流式嵌套布局



3.2 移动JavaScript框架

2013年移动互联网已经成为了各大互联网公司争先抢夺的入口，移动互联网的商机正在被挖掘。本节介绍目前比较典型的移动JavaScript框架，并帮助读者了解和使用这些移动JavaScript框架，帮助用户快速从PC的开发转移到移动开发中。这些框架包括Sencha Touch、jQuery Mobile、PhoneGap、JQ.Mobi等，最后会对各个移动JavaScript框架进行对比。

3.2.1 Sencha Touch

Sencha Touch是一款高性能的专门为移动设备开发的HTML 5移动应用开发框架，从ExtJS整合而来。使用Sencha Touch可以创建非常类似于Native App的Web App。Sencha Touch目前支持iOS、Android、BlackBerry、Windows Phone等主流移动应用开发。



Native App是一种基于智能手机本地操作系统如iOS、Android、Windows Phone，并使用原生程式编写运行的第三方应用程序。

Sencha Touch的特点如下：

- 超过50个内置组件。
- 拥有流行的主题。
- 内置MVC系统。
- 完全基于HTML 5技术，流畅的动画和无缝的滚动使得Sencha Touch看起来和一个本

地应用一样。

- 响应式布局，支持设备横屏和竖屏。

创建Sencha Touch应用相对比较简单，下面来创建一个查看当前天气的APP应用。

1. 导入Sencha Touch对应的JavaScript和CSS资源文件



Sencha Touch下载请前往官方网站http://www.sencha.com/products/touch/download/。

（1）在localhost根目录下创建weather文件夹，并把下载好的sencha-touch.css和sencha-touch-all-debug.js文件放到该文件夹下。

（2）创建index.html文件，代码如下：

```
<!DOCTYPE html>
<html>
<head>
      <meta charset="UTF-8">
      <title>天气预报</title>
      <link rel="stylesheet" href="sencha-touch.css" type="text/css">
             <!-- Sencha Touch样式 -->
      <link rel="stylesheet" href="app.css" type="text/css">
             <!-- 业务应用样式 -->
      <script type="text/javascript" src="sencha-touch-all-debug.js">
</script>        <!-- Sencha Touch脚本 -->
      <script type="text/javascript" src="app.js"></script>
             <!-- 业务应用脚本 -->
</head>
<body></body>
</html>
```

2. 创建app.js应用文件

```
Ext.YQL = {
    useAllPublicTables: true,
    yqlUrl: 'http://query.yahooapis.com/v1/public/yql',
    request: function(config) {
        var params = config.params || {};
        params.q = config.query;
        params.format = 'json';                  // 数据为json格式
        if (this.useAllPublicTables) {
            params.env = 'store://datatables.org/alltableswithkeys';
        }
        Ext.data.JsonP.request({                 // 远程jsonp请求
            url: this.yqlUrl,                    // 请求地址
            callbackKey: 'callback',             // 回调参数名
            params: params,                      // 传递参数
            callback: config.callback,           // 回调方法
            scope: config.scope || window        // 回调上下文
        });
    }
};
```

```
var tpl = new Ext.XTemplate(                          // 显示模板
    '<div class="weather">',
        '<h2>当前城市：{cityName}，温度：{code}&deg;</h1>',
        '<h3>查询时间：{time}</h3>',
    '</div>'
);
var makeYqlRequest = function(button) {
     Ext.YQL.request({
        query: 'select * from weather.forecast where woeid = 2132574', // 查询杭州的天气
        callback: function(success, response) {      // 查询结果返回
            var results = [];
            if (response && response.query && response.query.results) {// 获取返回值
                results = response.query.results;
                var weather_html = tpl.apply({       // 更新模板数据
                    cityName: results.channel.location.city,
                    code: results.channel.item.condition.code,
                    time: results.channel.item.condition.date
                });
                Ext.getCmp('weatherContent').setHtml(weather_html); // 根据id获取天气面板，更新数据
            }
        }
    });
}();
Ext.application({
    name: '天气预报',
    launch: function() {
        Ext.create("Ext.tab.Panel", {
            fullscreen: true,                        // 是否全屏
            tabBarPosition: 'bottom',
            items: [
                {`
                    xtype: 'nestedlist',             // 组件类型
                    title: '杭州天气',                // 标题
                    iconCls: 'home',                 // 图标样式
                    id: 'weatherContent',            // 元素id
                    html: 'Loading...'               // 加载提示
                },
                {
                    xtype: 'nestedlist', title: '关于', iconCls: 'info', displayField: 'title', cls: 'home',
                    html: [
                        '<div class="info">',
                        '<h1>数据来源</h1>',
                        "<p>天气数据来源于Yahoo (http://query.yahooapis.com) </p>",
                        '</div>'
                    ].join("")
                }
```

```
                ]
            });
        }
});
```

3. 创建内容样式文件

```
.weather, .info{                                    // 面板样式
      background: #f3f8ff;
      -webkit-border-radius: 2em;
      border-radius: 2em;
      -webkit-box-shadow: inset 0 1px 1px #bdd0ea;
      box-shadow: inset 0 1px 1px #bdd0ea;
      padding: 1em;
      text-align: center;
      text-shadow: 0 1px 0 #fff;
}
.weather h2, .info h1 {                             // 一级、二级标题样式
      color: #1c3961;
      font-size: 3em;
      font-weight: bold;
      margin-bottom: .1em;
}
.weather h3, .info p {                              // 三级和内容文字样式
      color: #53719b;
      font-size: 1em;
}
```

4. 运行文件

在手机浏览器或者PC上的Chrome浏览器打开该文件，即可直接运行。本示例在iPhone 4s上运行气象信息，效果如图3.4所示。



图3.4 iPhone 4气象信息示例运行效果

3.2.2 jQuery Mobile介绍和例子

jQuery Mobile是一个基于HTML 5技术搭建的一套触控友好的用户界面（UI）框架。该框架允许不编写任何JavaScript代码而让网站和应用可以运行在所有流行的移动设备平台上。其轻量极的代码架构，使其可更灵活地扩展和更易主题化设计。jQuery Mobile的基本特性主要包括以下几项。

- 简单性：框架使用简单，无须编写JavaScript或者极少编写JavaScript代码，就可以实现良好的UI效果和交互体验。
- 持续增强和优雅降级：jQuery Mobile构建于jQuery内核，不仅提供高端设备良好的用户体验，也支持低端设备，并尽可能的提供良好体验。
- 可访问性：jQuery Mobile在设计时考虑了用户的访问能力，它拥有Accessible Rich Internet Applications（WAI-ARIA）支持，以帮助使用辅助技术的残障人士访问Web页面。
- 主题设置：jQuery Mobile提供主题系统，允许提供自己的应用程序样式。

创建jQuery Mobile的应用很容易，下面来创建新闻阅读列表应用。

（1）创建一个HTML页面，命名为home.html，并在head标签中引入jQuery Mobile的库文件，代码示例如下：

```
<!doctype html>
<html>
<head>
      <title>阅读列表</title>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width,
initial-scale=1">
      <link rel="stylesheet" href="http://code.jQuery.com/mobile/1.3.2/
jQuery.mobile-1.3.2.min.css">
      <script src="http://code.jQuery.com/jQuery-1.9.1.min.js">
</script>
      <script src="http://code.jQuery.com/mobile/1.3.2/jQuery.mobile-
1.3.2.min.js"></script>
</head>
<body>
</body>
</html>
```

本示例使用的jQury Mobile的JavaScript和CSS文件直接引用了jQuery的CDN服务器，也可以直接从其官方网站下载到自己的服务器上。

（2）在body中添加第一个页面，代码如下：

```
<div data-role="page">
      <div data-role="header">                    <!-- 页头区域 -->
             <h1>阅读列表</h1>
      </div>
      <div data-role="content">                   <!-- 主显示内容区域 -->
             <ul data-role="listview" data-inset="true"
```

```
    data-filter="true">
07                          <li><a href="news.html" data-transition="slide"
    data-inline="true">关于公历的那些事</a></li>
08                          <li><a href="#">Tessel：用JavaScript做嵌入式开发
    </a></li>
09                          <li><a href="#">普通与顶级的UI设计师区别在哪里？</a></li>
10                          <li><a href="#">Linux 的启动流程</a></li>
11                          <li><a href="#">每个开发团队都该搞个git </a></li>
12                  </ul>
13          </div>
14          <div data-role="footer" data-position="fixed">  <!-- 页脚区域 -->
15                  <div data-role="navbar">                <!-- 导航区域 -->
16                          <ul>
17                                  <li><a href="#" data-icon="grid" class=
    "ui-btn-active">Home</a></li>
18                                  <li><a href="about.html"
    data-transition="flip" data-icon="about">About</a></li>
19                          </ul>
20                  </div>
21                  </div>
22 </div>
```

这个页面非常简单，主要包括三个部分，分别是头部（Header）、主体内容（Content）、尾部（Footer），jQuery Mobile的data-role是最主要的页面布局和UI渲染，通常有以下几项。

- 标识一个用页面：data-role="page"。
- 标识头部：data-role="header"。
- 标识主体内容：data-role="content"。
- 标识尾部：data-role="footer"。
- 标识为一个列表：data-role="listview"。

主体内容主要是一个ul结构的新闻列表，要显示为一个列表结构，只要指定ul元素上的data-role="listview"即可。

第07行中，指定a链接地址为new.html，指定切换效果为滑动（Slide）。在jQuery Mobile中，页面间的切换效果可以使用data-transition表指定。



更多的效果请参考http://jQuerymobile.com/demos/1.3.0/docs/widgets/transitions/。

第14行，设置data-position="fixed"，jQuery Mobile能识别它并把该底部导航固定在屏幕下方。

代码第15行，因为网页底部为一个navbar的导航，只需要在ul结构上指定data-role="navbar"就可以把ul结构渲染为底部标签进行切换。



有关更多jQuery Mobile的内容和使用方法请查看其官方网站http://jQuerymobile.com/的内容。

读者可以打开代码光盘，运行该示例，在浏览器中打开home.html，就可以看到一个非常简单的阅读列表，如图3.5所示。



图3.5 阅读列表

3.2.3 PhoneGap

PhoneGap是一个基于HTML、CSS和JavaScript的创建移动跨平台移动应用程序的快速开发平台。它使开发者能够利用iPhone、Android、Palm、Symbian、WP7、Bada和Blackberry智能手机的核心功能，包括地理定位、联系人、声音和振动等，此外PhoneGap拥有丰富的插件，可以以此扩展无限的功能。PhoneGap是免费的，但是需要特定平台提供的附加软件，例如iPhone的iPhone SDK、Android的Android SDK等，也可以和Dreamweaver 5.5配套开发。使用PhoneGap只比为每个平台分别建立应用程序好一点，因为虽然基本代码是一样的，但是仍然需要为每个平台分别编译应用程序。

电脑软件公司Adobe 2011年10月4日宣布收购创建了HTML 5移动应用框架PhoneGap和PhoneGap Build的新创公司Nitobi Software。Adobe表示收购PhoneGap后，开发者便可选择在PhoneGap平台使用HTML、CSS和JavaScript创建移动应用程序，也可选择使用Adobe Air和Flash。

安装PhoneGap非常简单，下面以在Mac平台下开发iOS应用为例，简单描述其应用的部署和开发实例。

1．安装Xcode

使用命令行工具创建iOS平台上的PhoneGap应用项目，首先需要安装Apple的开发工具XCode， Xcode的安装可以直接通过App Store来安装。



安装Xcode地址https://itunes.apple.com/us/app/xcode/id497799835?mt=12。

2．在XCode中安装Apple的Command Line Tools

Xcode安装完成后，PhoneGap要正常运行，需要为Xcode安装Command Line Tools，具体的安装方法为：打开Xcode应用程序，在Xcode下拉菜单栏中选择Preferences（属性）菜单，在弹出的属性菜单中选择Downloads（下载）选项卡，在下载选项卡中选择Components面

板，在Components的安装列表中找到Command Line Tools，单击Install按钮进行安装，如图3.6所示。

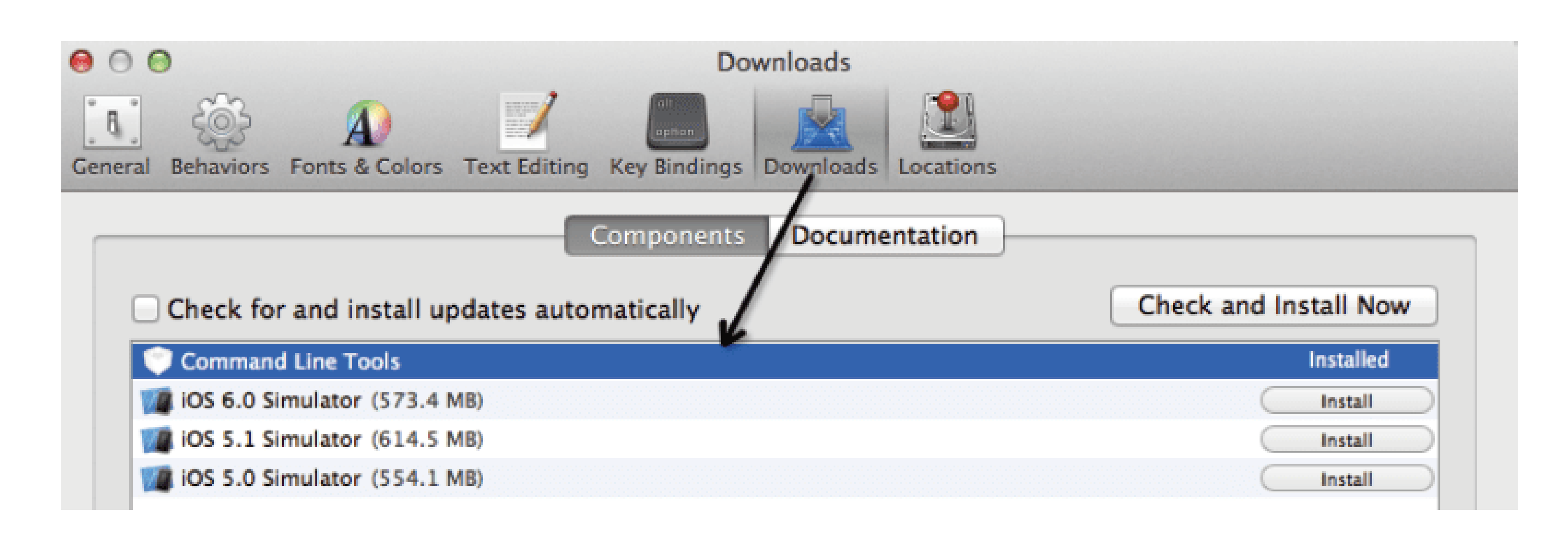


图3.6 在Xcode中安装Commane Line Tools

3. 安装PhoneGap

PhoneGap 3.0版本的安装只需要通过NPM包管理工具就可以，所以在安装PhoneGap之前请先确保安装Node.js。打开Shell控制面板，在命令行中，输入以下命令安装PhoneGap：

```
sudo npm install -g phonegap
```



有关Node.js的安装请参考本书第4章环境搭建的内容。

4. 安装Apache Cordova

Apache Cordova是用于构建、部署和管理基于HTML、CSS和JavaScript的命令行工具，目前支持Android、BlackBerry 10、iOS、Windows Phone 7 & 8等平台。在命令行中输入安装Cordova命令：

```
sudo npm install -g cordova
```

5. 创建最简单的Hello Word应用

在Shell中输入以下命令，创建一个HelloWorld应用程序：

```
cordova create hello com.example.hello "HelloWorld"
cordova platform add ios
cordova prepare
```

输入这三个命令后，一个iOS的PhoneGap应用程序已经创建完成，其目录结构如图3.7所示。

双击运行该目录结构下的“platforms/ios/HelloWorld.xcodeproj”文件，在Xcode中单击Run按钮，直接运行该示例程序，其结构如图3.8所示。

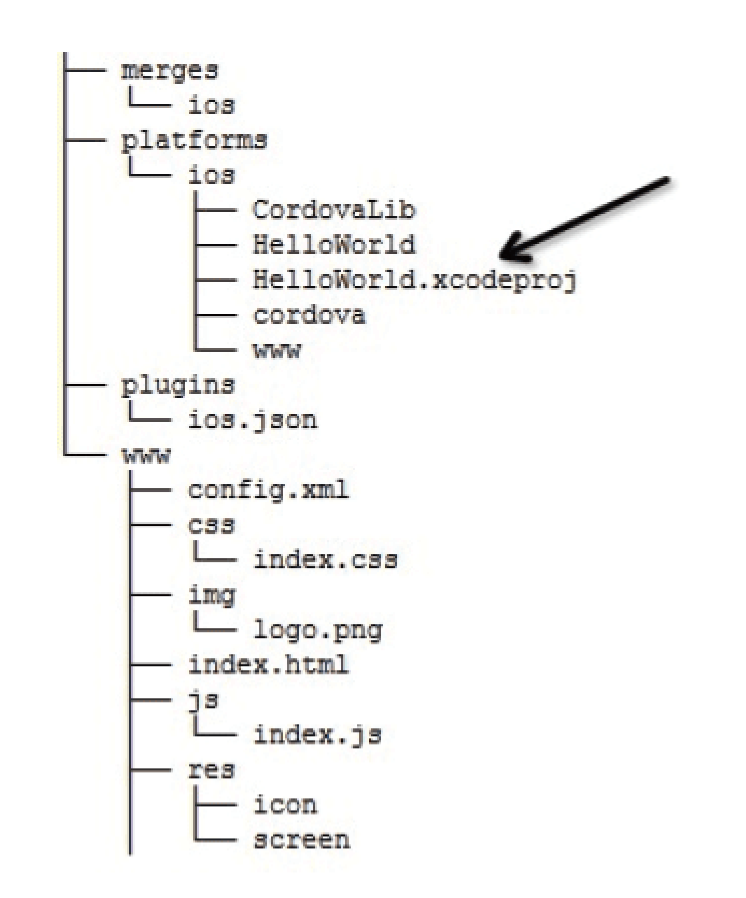


图3.7 PhoneGap在iOS平台上的目录结构



图3.8 运行结果



读者可以简单修改index.html文件的内容，然后再单击Run按钮，重新运行该程序。

3.2.4 JQ.Mobi

JQ.Mobi是HTML 5的移动框架，由Inter公司赞助。JQ.Mobi是jQuery的部分重写版本，所以其API和jQuery是一模一样的，使用起来非常容易上手。JQ.Mobi针对移动设备只提供必要的API，构成一个极快速的查询库，目前已经超过了60个API可以使用，而其文件大小只有16KB。

JQ.Mobi由三个组件组成。

- 快速查询：支持W3C，支持Windows 8JS。
- UI：提供一套友好的CSS程序库，完美支持原生的样式主题，包括iOS、iOS7、Android、Windows 8/WP8和Blackberry 10。
- JQ.Plugin插件：提供JQ.Plugin插件，以可以使用jQuery和UI库。同时也支持jQuery插件的书写模式，为jQuery应用开发者提供了便利。

下面简单描述如何使用JQ.Mobi开发移动应用。

1. 在页面中引入JS

```
<!DOCTYPE html>
<html>
    <head>
        <title>UI Starter</title>
        <meta http-equiv="Content-type" content="text/html;
charset=utf-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1">
        <meta name="apple-mobile-web-app-capable" content="yes" />
        <link rel="stylesheet" type="text/css" href="http://cdn.app-
framework-software.intel.com/2.0/icons.css" 09/>
```

```
        <link rel="stylesheet" type="text/css" href="http://cdn.app-
framework-software.intel.com/2.0/af.ui.css" />
        <script type="text/javascript" charset="utf-8"
src="http://cdn.app-framework-software.intel.com/2.0/appframework.min.
js"></script>
        <script type="text/javascript" charset="utf-8"
src="http://cdn.app-framework-software.intel.com/2.0/appframework.
ui.min.js"></script>
    </head>
    <body>
    </body>
</html>
```



相关的JS和CSS文件可以直接引用其提供的CDN地址，也可以从GitHub上下载。CDN地址为http://app-framework-software.intel.com/cdn.php，GitHub地址为https://github.com/01org/appframework。

2. 在body中搭建最基本的DOM结构

```
<div id="afui"> <!-- 这是最主要的容器，指定ID为afui，会自动加载AFUI主题 -->
    <!-- 这是顶部导航 -->
    <div id="header">
        <a href="javascript:$.ui.toggleSideMenu()" class="button"
style="float:right">侧边导航</a>
    </div>
    <!-- 这是内容区域 -->
    <div id="content">
        <!-- 这是一些页面面板，只需要添加class为pannel，如果设置为
selected="true"表示这是起始页面，每一个窗口只能显示一个面板 -->
        <div title='首页' id="home" class="panel" selected="true">这里
是首页内容</div>
        <div title='js代码' id="js" class="panel">这里是js页面</div>
    </div>
    <!-- 这是底部导航，以id="navbar"来标识 -->
    <div id="navbar">
        <div class="horzRule"></div>
        <a href="#home" id='navbar_home' class='icon home'>首页</a>
        <a href="#js" id='navbar_js' class='icon js'>JS页面</a>
    </div>
    <!-- 这是侧边导航，如果不需要可以去掉，以nav标签来标识 -->
    <nav>
        <div class='title'>侧边导航</div>
        <ul>
            <li >
                <a class="icon home" href="#home">首页</a>
            </li>
            <li >
                <a class="icon js" href="#js">JS页面</a>
            </li>
        </ul>
```

```
        </nav>
    </div>
```

3．运行

把步骤1、步骤2中的代码合并在一起就可以构建一个最基本的应用，在iOS手机浏览器运行以上代码，效果如图3.9所示。



图3.9 JQ.Mobi运行结果



3.3 CSS 3 UI框架

IE 6的市场份额在不断地减少，越来越多的开发者投奔CSS 3，导致近年来CSS 3 UI框架数不胜数。这其中也有一些非常优秀的框架，如Normalize.css、Bootstrap、uikit、Foundation、HTML 5 Boilerplate、HTML KickStart、Less Framework等。本节主要介绍HTML 5 Boilerplate和Less Framework。

3.3.1 HTML 5 Boilerplate

HTML 5 Boilerplate是一个集成了非常多先进特性的前端模板框架，能够帮助开发者快速构建健壮的、响应式的Web应用和站点。

HTML 5 Boilerplate的核心是帮助开发Web应用和站点，包括以下几项。

- 一个简洁而友好的移动HTML模板，Google分析异步化，提供触摸设备的图标集合以及其他的几十个额外的窍门和技巧。
- 引入Normalize.css库，Normalize.css是一个流行的HTML 5重置样式库，包括基础样式

重置、媒体查询和打印样式重置等等。

- 引入Modernizr库，自动检测所有的浏览器是否支持HTML 5的新特性，对于不支持的浏览器将提供优雅降级。
- 高性能，HTML 5 Boilerplate非常注重性能，其提供了一套Apache的配置文件，支持JavaScript和CSS文件静态化，提供多种方式的脚本压缩的打包方式，包括使用Grunt和Ant等。

HTML 5 Boilerplate的官网地址为http://HTML 5boilerplate.com/，目前最新版本为4.3.0，读者可以前往官方下载，如图3.10所示。

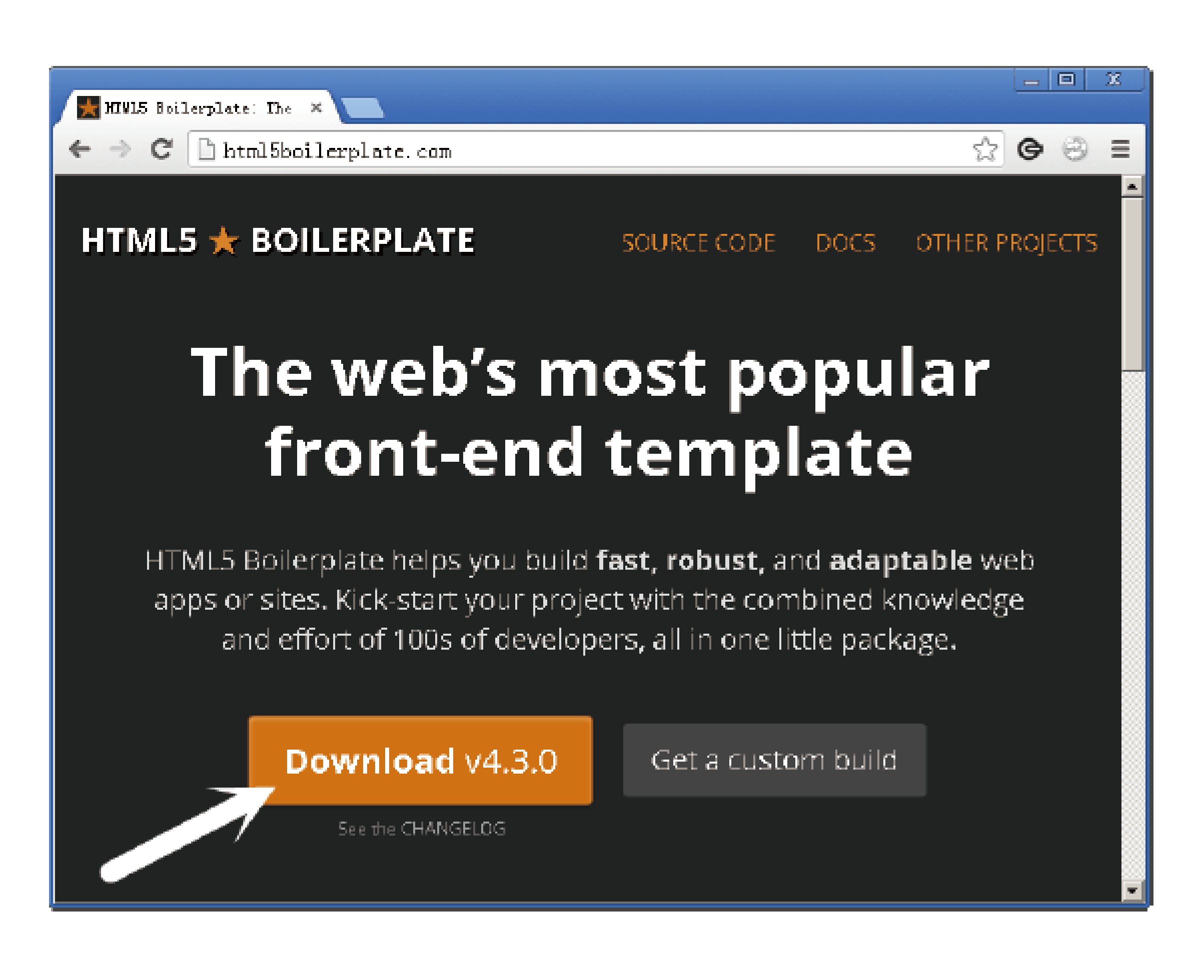


图3.10 HTML 5 Boilerplate官方下载

3.3.2 Less Framework

Less Framework是一个自适应的CSS栅格系统，提供4种布局方式和三套预设的排版。4种布局方式分别为以下几项。

- 默认布局：这是一套默认的布局方式，992像素宽度，共10个栅格。对于台式机、笔记本电脑、横向的平板电脑及老式的浏览器，按黄金比例左右分割，左侧6栏，右侧4栏。
- 平板布局：将768px的屏幕分辨率分成8栏，主要用于iPad和其他一些平板电脑。对于一些长文本，直接分割成6栏并居中显示。一些较短的文本，则分割成左右两列，分别占4栏。
- 移动设备布局：320px的移动平台，如iPhone、iPod Touch和其他一些流行的移动设备，将320px分割成三栏。
- 宽版移动布局：对于较大屏幕分辨率的移动设备，或者移动设备横向展示时，将采用480px的宽度，把布局分成5栏。

Less Framework的官网地址为http://lessframework.com/，读者可以前往官方下载，如图3.11所示。

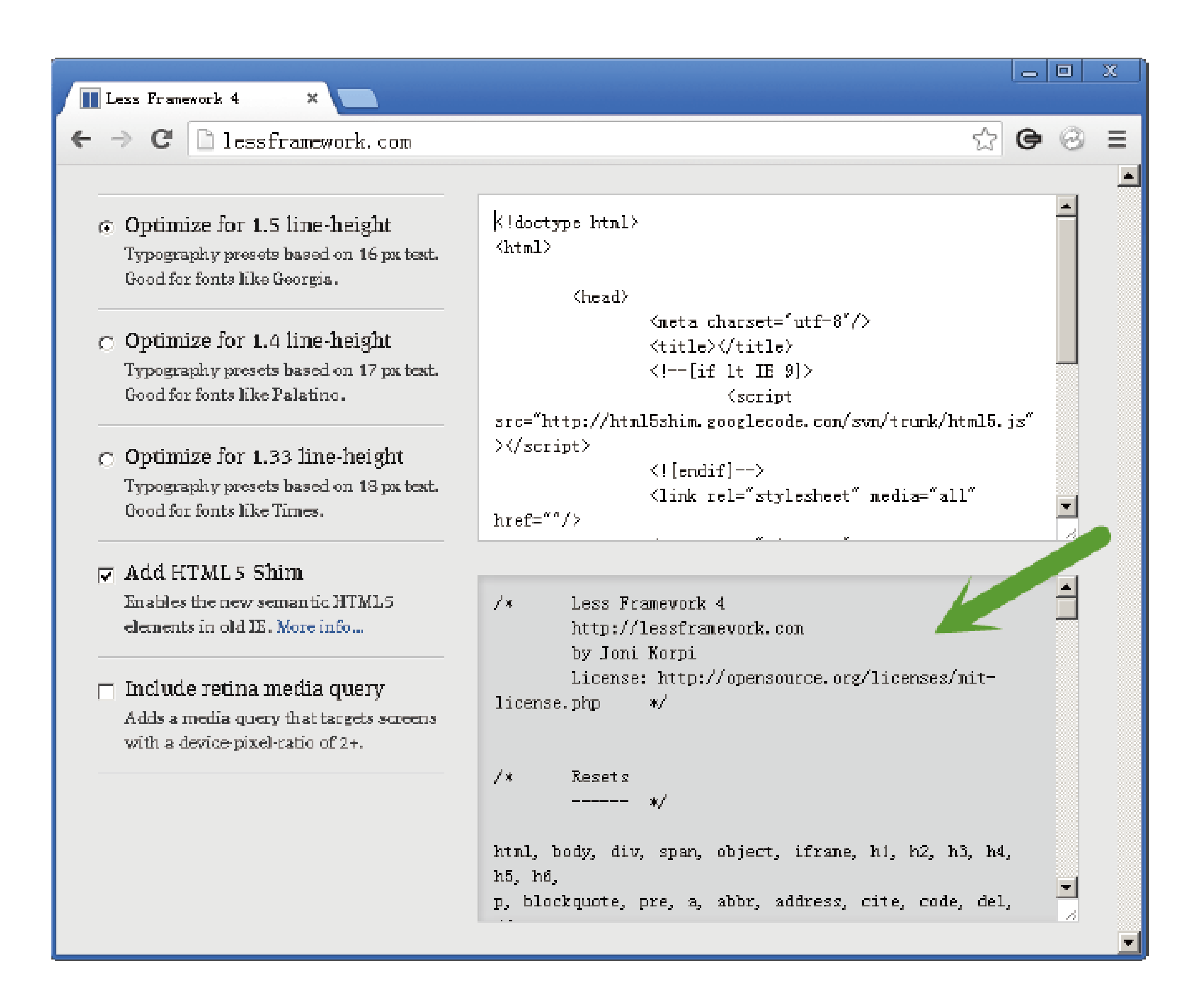


图3.11 Less Framework下载



3.4 HTML 5图表库

HTML 5图表库主要是使用HTML 5技术实现绘制图表的功能，主要包括曲线图、拆线图、饼图、环形图、面积图、柱形图和条形图等。目前HTML 5图表库非常多，大多数图表库所使用的技术都是采用HTML 5的Canvas和SVG，或者用这两者的结合来绘制图表。本节将通过两个非常典型的例子来说明一般的图表库的使用，第一个是Raphael图表库，第二个是鼎鼎大名的Highcharts。

3.4.1 Raphael

Raphael是一个小型的图表框架，很容易通过Web构建矢量图形。Raphael使用SVG和VML作为构建图形的基本技术，通过其对象，构建出DOM结构，因此也可以利用JavaScript在DOM结构上绑定事件。Raphael的目标是为了提供一个在Web非常容易绘制的矢量图形，并且很容易的运行在各个浏览器上。Raphael目前支持的浏览器有Firefox 3.0+、Safari 3.0+、Chrome 5.0+、Operating 9.5+、IE 6.0+。

以下是一个使用Raphael提供的path方法和animate方法连续画HTML 5这几个字母的例子，代码如下：

```
<!DOCTYPE html>
<html>
      <head>
             <title>Raphaël·画HTML 5字母</title>
```

```
            <link rel="stylesheet" href="demo.css" media="screen">
            <link rel="stylesheet" href="demo-print.css" media="print">
            <script src="raphael-min.js"></script>
<!-- raphael图形库压缩脚本 -->
            <style media="screen">
            body {background: #333;}
            #holder {
                   height: 480px; left: 50%; margin: -240px 0 0 -320px;
                   position: absolute; top: 50%; width: 640px;
            }
            </style>
      </head>
      <body><div id="holder"></div></body>
<script>
      var HTML5 = [                  // 定义HTML 5这几个字的绘制路径
      // HTML 5字母路径数据省略，读者可以参看本书网络资源
      ];
window.onload = function () {
      var r = Raphael("holder", 800, 600);
      // 新建Raphael实例，并设置宽高
      var h5 = r.path('M0,0L0,0z').attr({               // 回执一条路径
            fill: "#fff",                                // 填充色
            stroke: "#fff",                              // 边框色
            "fill-opacity": .3,                          // 填充色的透明度
            "stroke-width": 1,                           // 边框的宽度
            "stroke-linecap": "round",                   // 边框的线帽
            translation: "100 100"
      });
      var i = 0;
      var run = function(keyword){
            h5.animate({path: HTML 5[i]}, 200, 'linear', function(){
      // Raphael提供动画库
                   i++;
                   if(i <= 4){                         // 是否已经画完
                          setTimeout(run, 500);        // 暂停0.5s画下一个字母
                   }
            });
      }
      run();                                             // 执行逻辑脚本
};
</script>
</html>
```

- 定义数组：该示例首先定义HTML 5这个数组，该数组为HTML 5这5个字的矢量路径。利用animate方法可以把定义好的矢量路径按动画的方式绘制出来。
- path方法：path方法可以接收一个绘制路径和一组配置信息，在本示例中，先在画布的最左上角绘制一个点，并填充为白色，边框也为白色的1像素点。
- animate方法：该方法可接受多个参数，第1个参数可接收一组配置信息，本示例中为矢量路径，第二个参数为动画时间，在200ms内，把矢量路径画出。



有关Raphael的使用方法可查看其API文档，地址为http://raphaeljs.com/reference.html。

3.4.2 Highcharts

Highcharts与Raphael的最大区别在于，Highcharts是基于配置自动生成一个可视数据图表，Highcharts支持的图表类型非常多，包括直线图、曲线图、散状图、区域图、区域曲线图、柱状图、点状图、饼状图等等。Highcharts有几大特点。

- 提示功能：鼠标移动某个点后，可以根据配置信息进行相应的文字提示，并且该文字提示是完全可定制的。
- 放大功能：可以选中图表的某一个部分进行放大，以更清楚的状态查看图表。
- 组合与取消组合功能：可以在一个图表上显示不同维度的线图，可以方便地隐藏和显示不同维度的数据。
- 时间轴：支持三维的刻度表示。
- 后端兼容：容易与后端数据兼容，使用方便。

以下演示了一个网站的用户增长曲线图，完整代码请参考光盘目录3.4.2代码部分，关键脚本代码如下：

```
define(function(require, exports, module){
      var $ = require('jQuery'),                  // 引入JQuery类库模块
             venus = require('venus'),            // 引入venus类库模块
             ui = require('ui'),                  // 引入ui类库模块
             container = document.getElementById("container");
      var svg = {
             lineData: [],                        // x、y轴名称
             timeDiff: {},                        // 获取数据的时间段
             lineOption : {
                    width: 700,
                    height: 400,
                    axis: {                       // 坐标系x、y轴数据
                           x: {
                                  opposite: false,     // 是否跟默认位置相反
                                  tickWidth: 70,
      // 设置x轴坐标点是否出现占位符号和其宽度
                                  ticks: ['1', '2', '3', '4', '5',
 '6',7,8,9,10]
                           },
                           y: {
                                  min: 0               // y轴最小值
                           }
                    },
                    icons: {
                           0: 'circle'                 // 间隔点图形
                    },
                    line: {
                           smooth: true,               // 曲线是否平滑
```

```
                dotRadius: 5                // 曲度
            },
            grid: {}
        },
        _getData : function(callback){            // 从后台服务获取数据
            var self = this;
            $.post("api_get_user_total.php", self.timeDiff, function(res){
                var data = $.parseJSON(res);        // 将数据转为JSON格式
                callback && callback(data);
            });
        },
        _setData: function(data){                 // 填充图标数据
            var self = this, arr = [], time = [];
            data.forEach(function(l, i){
                time.push(l.time);
                arr.push(l.total);
            });
            self.lineData = [{name: 0, data: arr}];
            self.lineOption.width = arr.length * 70;
            self.lineOption.axis.x.ticks = time;        // x轴数据
        },
        _show: function(){                        // 输出图形
            var self = this;
            container.innerHTML = '';             // 清空容器
            new Venus.SvgChart(container, self.lineData, self.lineOption);
        },
        run: function(){                          // 执行绘制
            var self = this,
            cb = function(data){
                self._setData(data);
                self._show();
            }
            this._getData(cb);
        },
        render: function(startTime, endTime){     // 绘制图形
            this.timeDiff.startTime = startTime || '';
            this.timeDiff.endTime = endTime || '';
            this.run();
        }
    };
    exports.svg = svg;                            // 设置模块对外接口
});
```

该程序运行效果如图3.12所示。

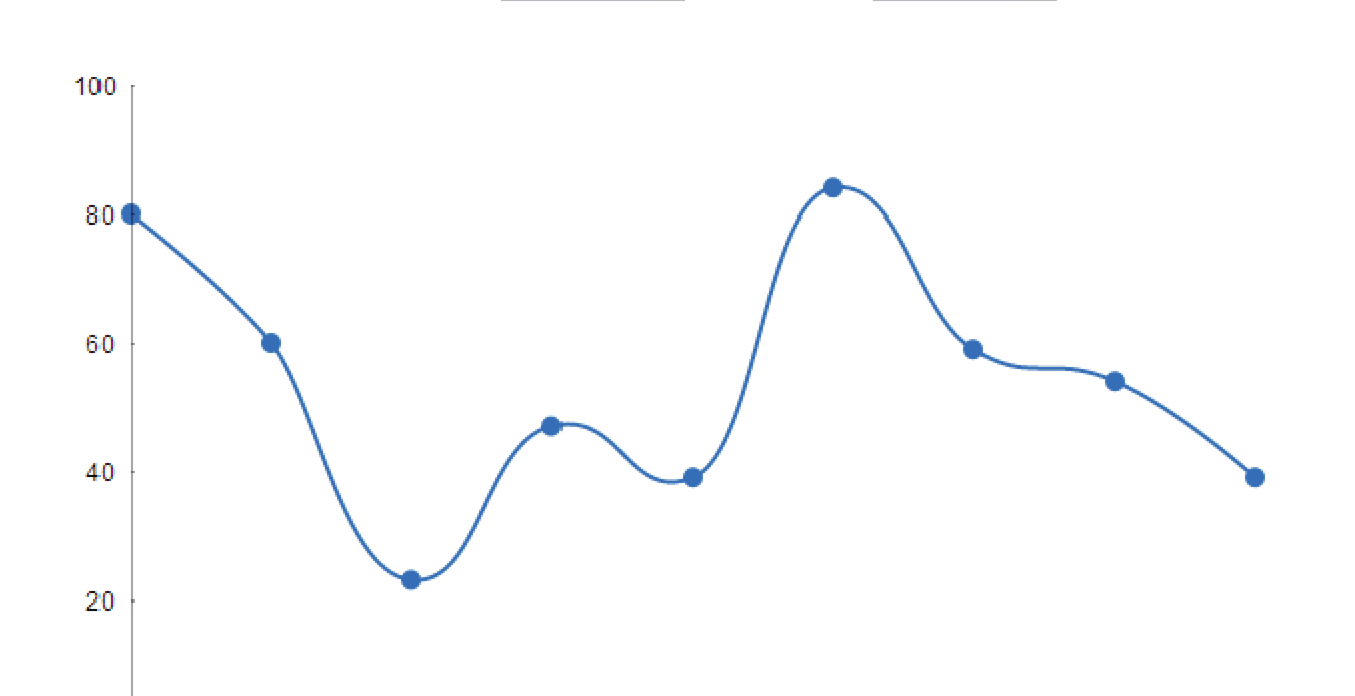


图3.12　用户增长曲线

3.5　游戏库——Three.js的使用

随着HTML 5的流行，使用JavaScript来写游戏逐渐变得可能。基于HTML 5而发展起来的游戏库也越来越多，如提供2-D模型的GMP游戏引擎、使用Canvos和DOM的Collie游戏库、以WebGL或Canvas作为渲染引擎的Three.js游戏库等等，这些游戏库有一个共同的特点，都是基于HTML 5的技术。本节主要通过Three.js这个游戏库来分析游戏引擎是怎么工作的。

Three.js是一个伟大的、开源的、以WebGL作为渲染引擎的JavaScript框架，利用HTML 5的Canvas可以绘制出3-D模型，使用Three.js可以在浏览器上可以实现真正的3D效果。使用Three.js分为6个基本步骤，分别是：

（1）设置渲染器。
（2）设置相机Camera。
（3）设置场景Scene。
（4）设置光源Light。
（5）设置物体Object。
（6）设置动画旋转。

以下是一个使用Three.js的例子，读者可以把代码保存为threejs-demo.html，在Chrome浏览器中运行。

```
<html>
      <head>
      <title>My first Three.js app</title>
      <style>canvas { width: 800px; height: 600px;}</style>
```

```
        </head>
        <body>
        <script src="https://rawgithub.com/mrdoob/three.js/master/build/three.js"></script>
        <script>
        // 1、设置渲染器
        var renderer = new THREE.WebGLRenderer({antialias:true});
        // 生成渲染器对象属性，锯齿效果设为true
        renderer.setSize(800, 600);       // 设置渲染器的宽和高，和画布大小一致
        document.body.appendChild(renderer.domElement);
        // 追加canvas元素到body元素当中
        renderer.setClearColor(0xFFFFFF, 1.0);          // 设置渲染器的清除色
        // 2、设置摄像机
        var camera = new THREE.PerspectiveCamera(45, 800/600, 1, 1000);
        //此处为设置透视投影的相机，默认情况下，相机的上方向为Y轴，右方向为X轴，沿着
        Z轴垂直朝里（视野角：fov； 纵横比：aspect； 相机离视体最近的距离：near；
        相机离视体最远距离：far）
        camera.position.x = 400;                // 设置相机的位置坐标
        camera.position.y = 0;
        camera.position.z = 0;
        camera.up.x = 0;
        camera.up.y = 1;                        // 设置相机上方向为y轴
        camera.up.z = 0;
        // 3、设置场景
        var scene = new THREE.Scene();
        // 4、设置光源
        var light = new THREE.DirectionalLight(0xFFFFFF, 1.0, 0);
        // 设置平行光DirectionalLight
        light.position.set(50, 50, 50);         // 光源向量，即光源的位置
        scene.add(light);                       // 追加光源到场景
        // 5、设置物体
        var geometry = new THREE.CubeGeometry(50, 50, 50);
        // 形状 （宽 高 深度）
        var material = new THREE.MeshLambertMaterial({color: 0Xff3300});
        // 材质
        var cube = new THREE.Mesh(geometry, material);
        scene.add(cube);
        // 6、设置动画旋转
        var t = 0;
        var render = function () {              // 动画绘制
              t++;
              renderer.clear();                 // 清空画布
              cube.rotation.set(0, t/100,0);
              camera.lookAt({x:0,y:0,z:0});
              renderer.render(scene, camera);   // 渲染物体和相机
              requestAnimationFrame(render);
        };
        render();
        renderer.clear();
        renderer.render(scene,camera);
        </script>
```

```
</body>
</html>
```

本例运行结果如图3.13所示。



图3.13　光源从左侧照到物体上的3D效果

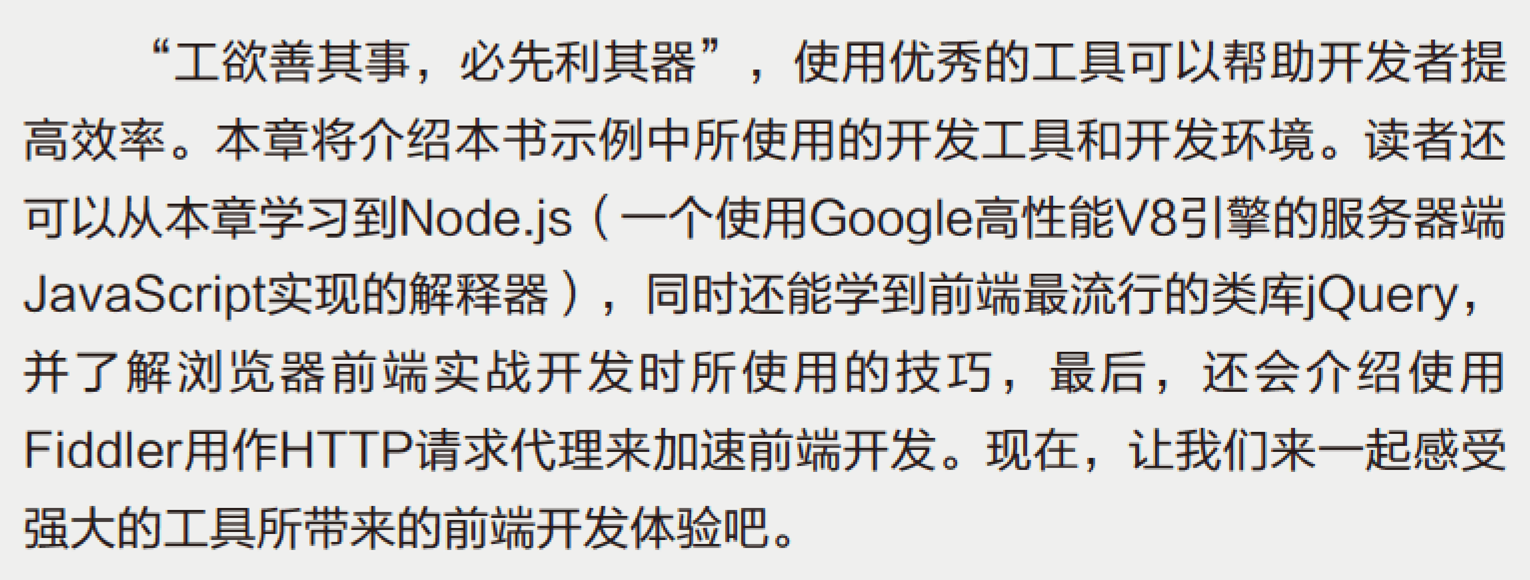


3.6 本章小结

本章主要介绍了HTML 5的一些概念、框架及其简单的使用。响应式网站的设计理念和设计哲学已经被越来越多的开发者所关注，一些较大的网站也已经实现了全站响应式，如“一淘”网。至于使用响应式设计是增加了成本还是降低了成本，也因网站和团队而异，因此是否使用响应式，需要根据自身团队情况进行衡量。2013年是移动互联网爆发的上升期，国内一些大网站也把移动互联网作为企业的重要战略之一，有些大企业甚至把移动互联网的地位看得比传统的PC还要重。随着HTML 5技术的成熟，越来越成熟的图表库、3D库、游戏库频频出现，使得前端开发人员可以玩转更多更酷的技术。



第 4 章 环境搭建



“工欲善其事，必先利其器”，使用优秀的工具可以帮助开发者提高效率。本章将介绍本书示例中所使用的开发工具和开发环境。读者还可以从本章学习到Node.js（一个使用Google高性能V8引擎的服务器端JavaScript实现的解释器），同时还能学到前端最流行的类库jQuery，并了解浏览器前端实战开发时所使用的技巧，最后，还会介绍使用Fiddler用作HTTP请求代理来加速前端开发。现在，让我们来一起感受强大的工具所带来的前端开发体验吧。

本章知识点：

- 适用HTML 5的编辑器
- 如何使用Node.js
- jQuery框架
- 使用Chrome浏览器调试脚本



4.1 选择一款编辑器

开发人员在从事前端开发工作之前，选择一款合适的编辑器是非常关键的一步。很多初学者，常常使用一些功能较重、体积巨大的开发工具作为前端编辑器，如Eclipse、Microsoft Visual Studio、Adobe Dreamweaver等。其实，现在市面上有很多体积轻巧，但功能非常强大的编辑器供选择，下面罗列一些比较优秀的适用于前端开发的编辑器供读者参考。

4.1.1 Notepad++编辑器

Notepad++是一款Windows环境下免费开源的代码编辑器，图4.1为Notepad++编辑器截图。

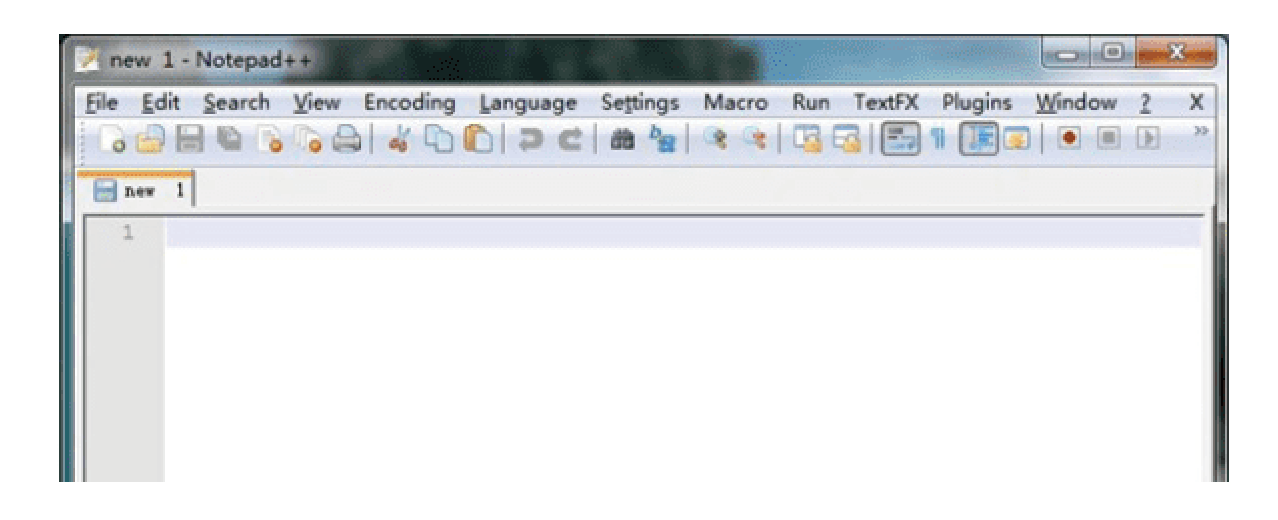


图4.1 Notepad++编辑器

Notepad++编辑器内置支持JavaScript、C++、Java等27语法高亮显示等功能，自动检测文件类型，根据语法节点自由折叠或者打开代码，还可以支持如宏功能、插件扩展等，项目托管于SourceForge.net之上。

4.1.2 UltraEdit编辑器

UltraEdit也是Windows的一款流行的老牌文本编辑器，同时该款编辑器被移植到Linux平台，名为UEX（全称UltraEdit for Linux），图4.2为UltraEdit编辑器截图。

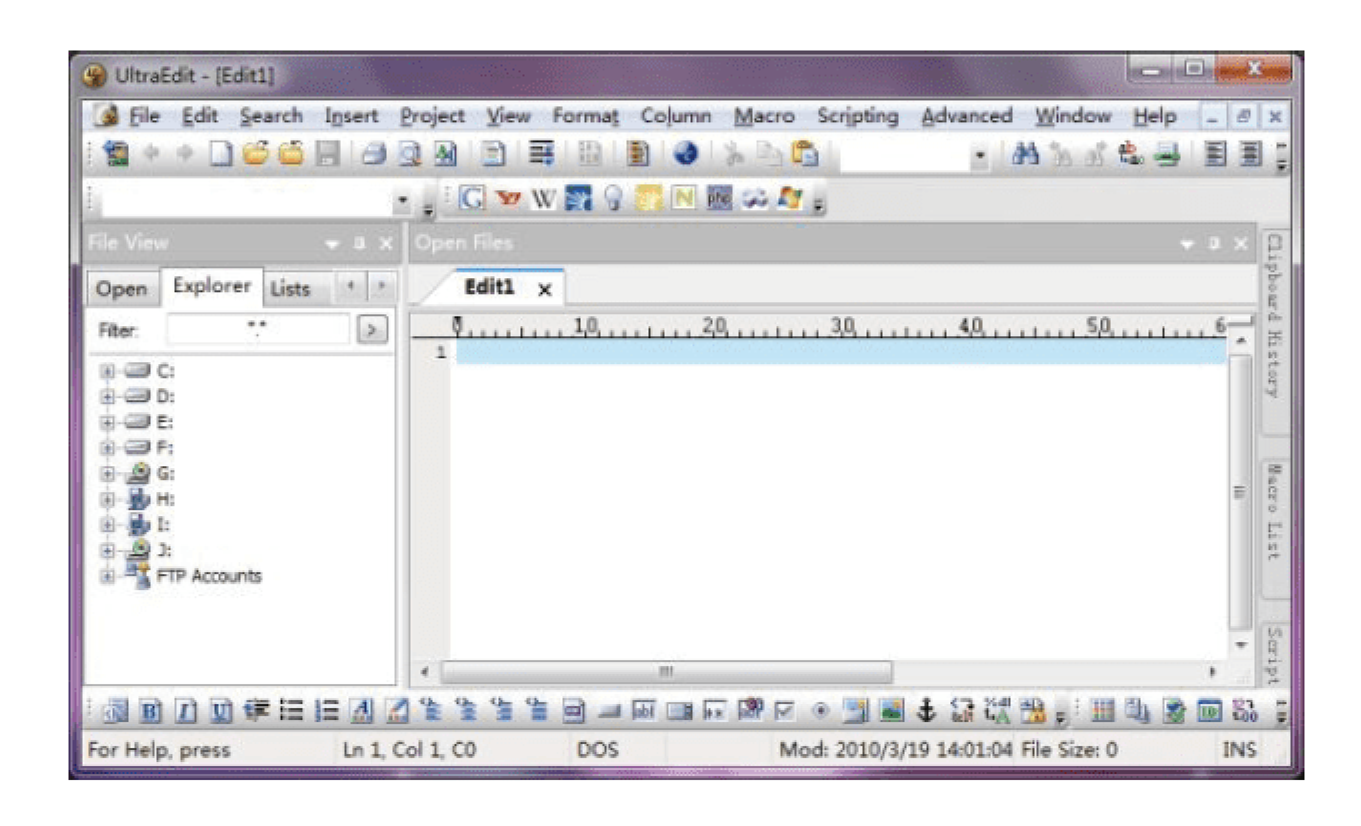


图4.2 UltraEdit编辑器

UltraEdit编辑器可以编辑文本、十六进制、ASCII 码，打开的文档大小不受限制，即使是数兆字节的文件也只占用最小限度内存，即便如此，该款编辑器文件打开速度还是飞快的，是一款非常优秀的编辑器。

4.1.3 Sublime Text 2编辑器

笔者目前使用的编辑器名为Sublime Text 2，除了拥有普通编辑器具备的语法高亮、代码提示补全、代码折叠、自定义皮肤、配色、多便签页的功能之外，还拥有非常棒的代码地图、多种界面布局与全屏免打扰模式功能，图4.3为Sublime Text 2编辑器的截图。

```
E:\hello.txt • - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
hello.txt

var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');


var net = require('net');

var server = net.createServer(function (socket) {
  socket.write('Echo server\r\n');
  socket.pipe(socket);
});

server.listen(1337, '127.0.0.1');

Line 16, Column 34                                Tab Size: 4        Plain Text
```

图4.3 Sublime Text 2编辑器

Sublime Text 2除了上面介绍的功能之外还有扩展包管理器、功能强大的命令面板、切换为VIM操作等，由于功能实在太多，不再过多介绍，下面列出该款编辑器最受欢迎的几点原因：

- 可随意转到任何地方。
- 多重选择。
- 命令调色板。
- 免打扰模式。
- 分离编辑。
- 即时专案开关。
- 插件API。
- 自定义任何东西。
- 跨平台（OS X、Windows、Linux）。

读者可以前往官网地址http://www.sublimetext.com/下载体验。



4.2 Node.js

Node.js自从被推出以来，就一直受到前端开发工程师的关注，因为开发者可以只使用一种语言JavaScript就能完成所有的前、后端的开发工作。下面罗列了整个Node.js迅速崛起的发展史：

- 2009年2月，Node.js作者Ryan Dahl在博客上宣布准备开始开发工作。
- 2009年5月，Ryan Dahl在GitHub上发布了最早的Node.js版本。
- 2009年11月，JSConf大会邀请Ryan Dahl做Node.js讲座。
- 2010年4月，JSConf大会再次邀请Ryan Dahl做Node.js讲座。
- 2010年底，Node.js获得云计算服务商Joyent资助。
- 2011年7月，Node.js在微软的支持下发布Windows版本。

接下来，一同感受Node.js所带来的新奇开发体验吧。

4.2.1 Node.js介绍

相信很多关心前端的读者，已经在大量的文章和技术周刊上看到关于Node.js的报道和新闻，那么首先要明确的是“Node.js到底是什么呢？”。Node.js是服务器端JavaScript的解释器，也就是说是可以通过JavaScript操作系统级别的功能。

按照官方的解释，Node.js是一个可以轻松构建和可扩展的网络应用平台，该平台基于Chrome的JavaScript运行环境。Node.js有如下几个特点：

- 事件驱动。

- 非阻塞I/O模型。
- 单线程。
- 基于V8引擎。

前面提到Node.js基于V8引擎，实际上是对V8引擎的封装和特定场景的优化，那么V8是什么？V8是谷歌用于Chrome浏览器的底层JavaScript的引擎，运行前将JavaScript编辑成机器代码，提高运行性能。

Node.js经过近几年的发展，已经有大批的优秀企业使用了该技术开发应用，如全球最大的职业社交网站LinkedIn使用Node.js作为移动服务器后台基础，使其应用中服务器通信性能大大提升，完胜基于Ruby on Rails技术方案。在国内，淘宝也使用Node.js开发了MyFOX，一个数据处理中间件，用于从一个MySQL集群中读取数据，并计算出统计结果返回输出。大家使用Node.js的理由都是基于自身的特点和卓越的性能表现，是否很有兴趣在日常开发的项目中使用Node.js？那就快行动起来，一同体验前后端统一开发语言的时代。

4.2.2 Node.js安装

通过前面的介绍，相信读者对Node.js已经有初步的了解，接下来，本节将介绍Windows环境下的Node.js安装。

首先，去Node.js官网（http://nodejs.org/）下载安装包，图4.4为官网下载页面。

Windows Installer node-v0.10.13-x86.msi	Macintosh Installer node-v0.10.13.pkg	Source Code node-v0.10.13.tar.gz
Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.exe)	32-bit	64-bit
Mac OS X Installer (.pkg)	Universal	
Mac OS X Binaries (.tar.gz)	32-bit	64-bit
Linux Binaries (.tar.gz)	32-bit	64-bit
SunOS Binaries (.tar.gz)	32-bit	64-bit
Source Code	node-v0.10.13.tar.gz	

图4.4 Node.js官方下载

这里按照笔者的机器环境配置，选中“Windows Installer (.msi) 64-bit”安装包，当前Node.js的版本是v0.10.12，下载后的是“node-v0.10.12-x64.msi”安装程序。

（1）双击该安装包，出现安装提醒界面，单击Next按钮，执行下一步安装操作，如图4.5所示。

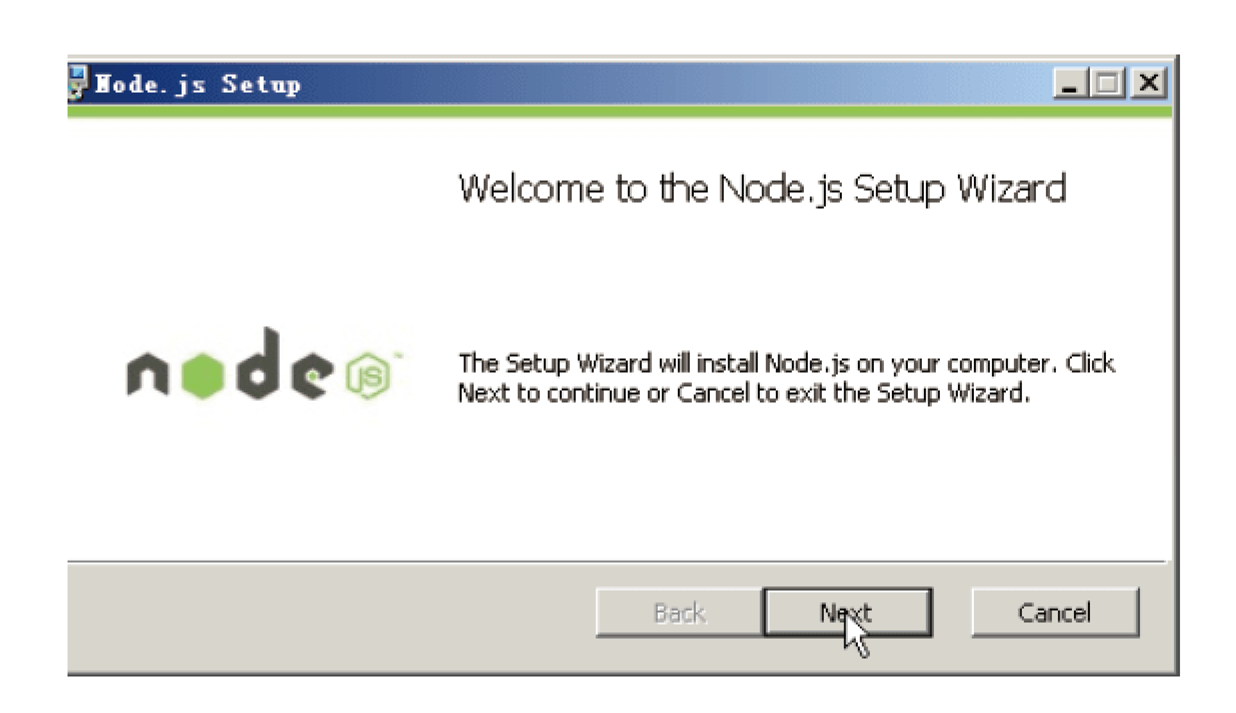


图4.5 欢迎界面

（2）选中I accept the terms in the License Agreement复选框，并单击Next按钮进入下一步，如图4.6所示。

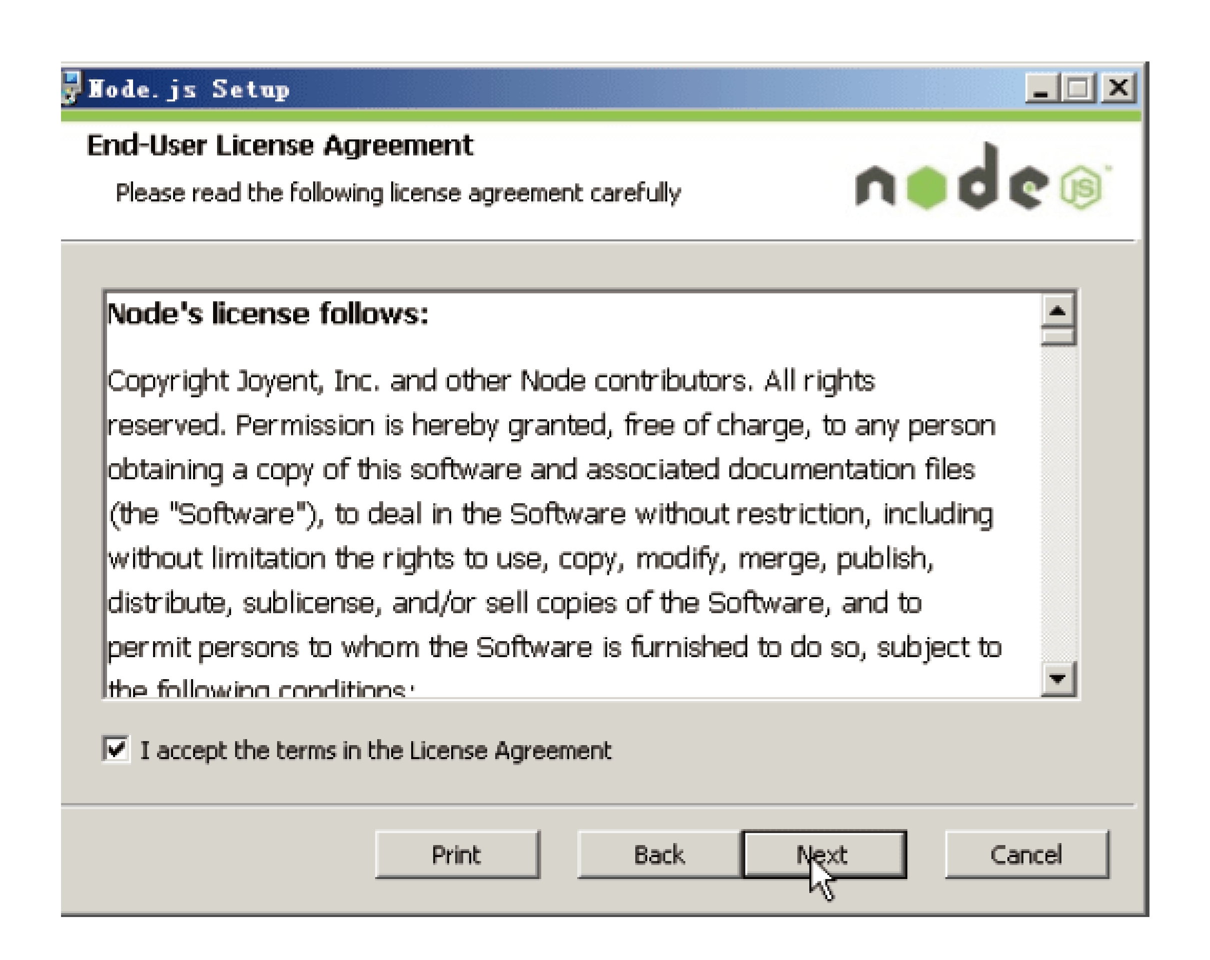


图4.6 安装协议

（3）默认安装目录为“C:\Program Files\nodejs\”，确认后继续单击Next按钮执行下一步，如图4.7所示。

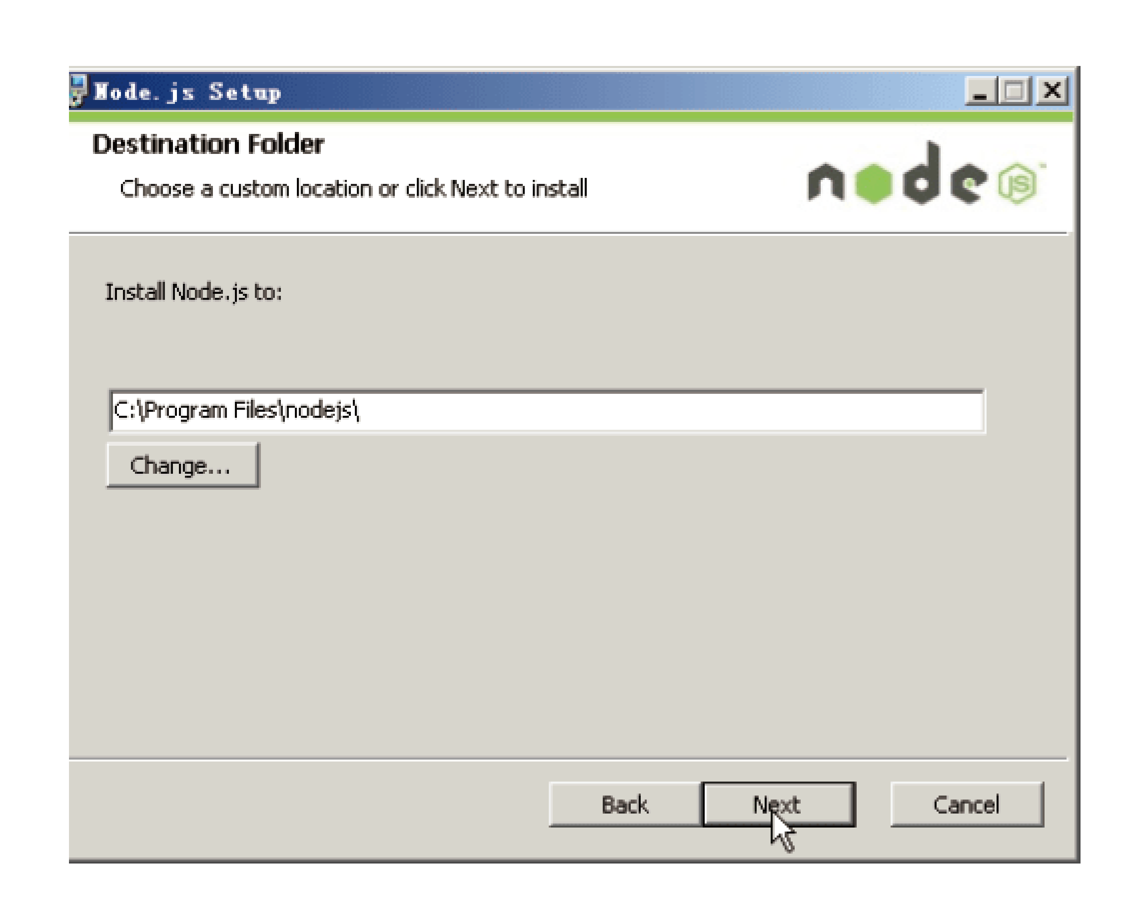


图4.7 确认安装目录

（4）进入自定义安装页面，默认为全部选中，其中Node Package Manager是Node.js的套件管理工具，后续会做进一步的介绍。继续单击Next按钮进入正式安装，如图4.8所示。

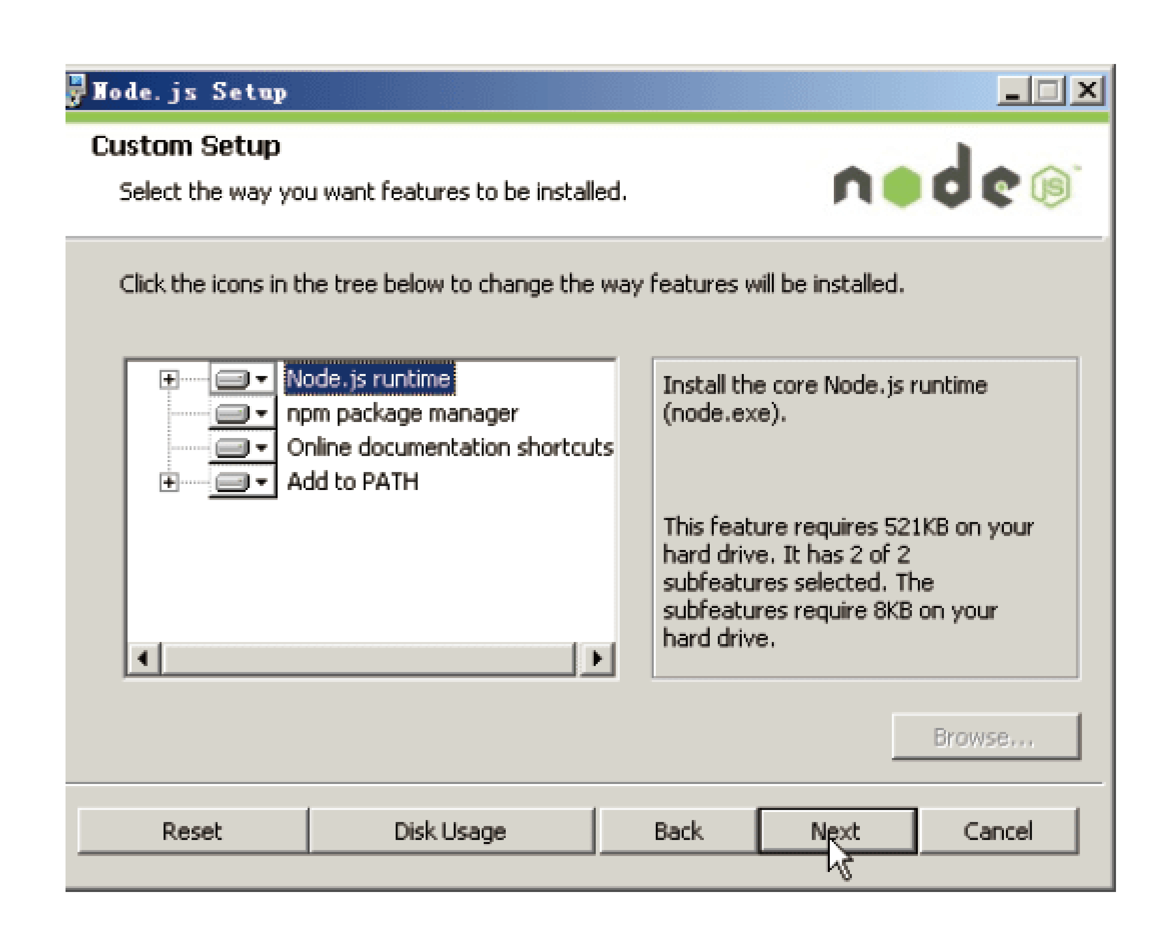


图4.8　自定义组件安装界面

（5）单击安装准备界面的Install按钮，开始安装Node.js，如图4.9所示。

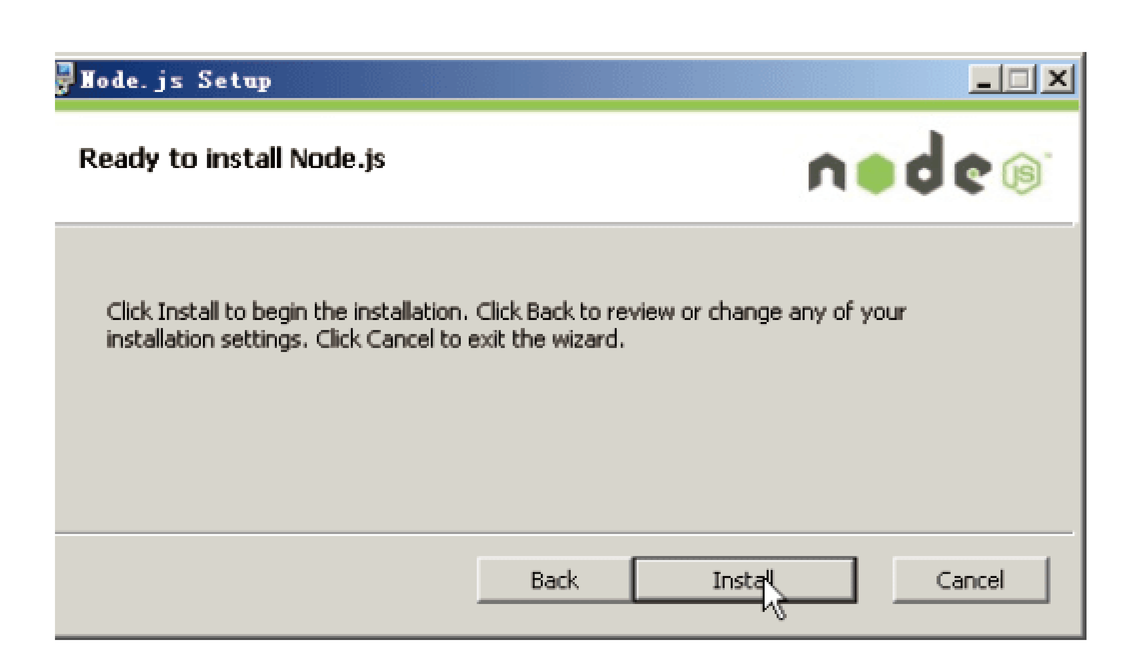


图4.9　安装准备界面

（6）安装程序开始进行Node.js的安装，如图4.10所示。

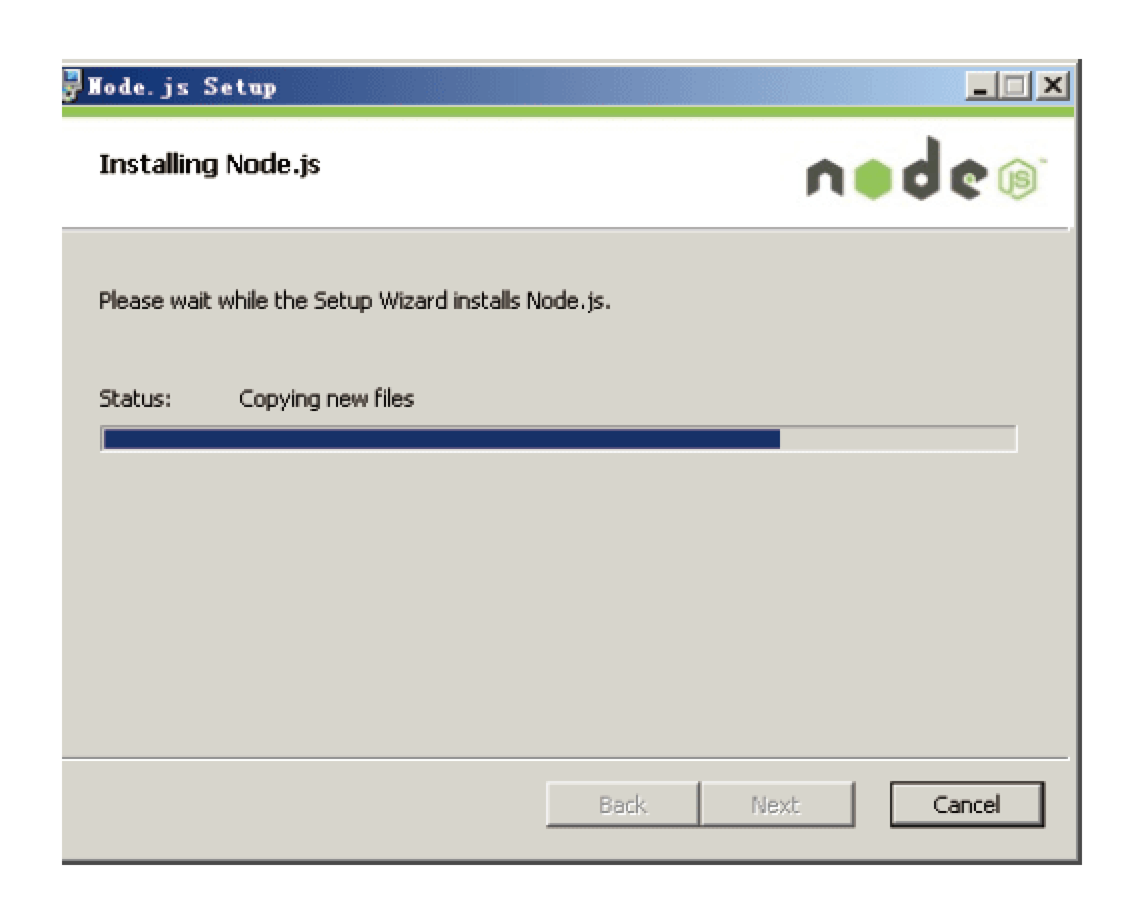


图4.10　正在安装Node.js

（7）安装完毕后，出现如图4.11所示的界面，表示安装成功。

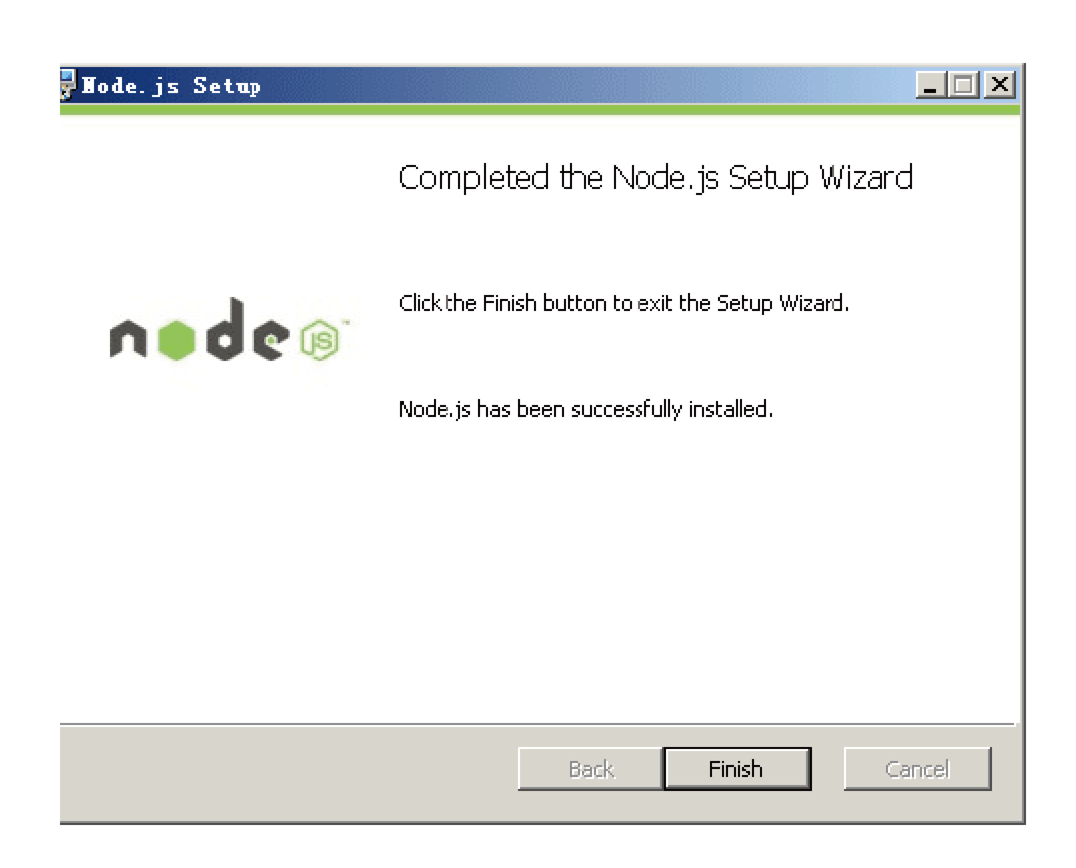


图4.11 Node.js安装成功

下面验证Node.js是否真的安装成功。先进入目录“C:\Program Files\nodejs”，新建文件“hello world.js”，代码如下：

```
console.log("hello world");
```

打开Windows命令窗口，进入“C:\Program Files\nodejs”目录，执行代码“node "hello world.js"”，输出效果如图4.12所示。



图4.12 “hello world.js”示例输出效果

确认Node.js已经安装成功，读者如果觉得示例代码不过瘾，可以使用官网提供的示例进一步体验Node.js开发的便捷。

4.2.3 使用Node.js的NPM

NPM全称Node Package Manager，是Node.js的模块管理工具，通过使用NPM可以安装或者卸载共享在网络上的程序包，同时个人也可以发布自己的程序包到NPM供他人使用。

在使用之前，首先检查NPM是否正确安装，打开Windows命令提示工具，执行代码如下：

```
npm -v                                    // 或者npm --version
```

出现图4.13，表示NPM安装成功，笔者当前的NPW版本号是1.1.45。

```
C:\WINDOWS.0\system32\cmd.exe
Microsoft Windows [版本 5.2.3790]
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>npm -v
1.1.45
```

图4.13 NPM安装成功

NPM目前拥有35 631个程序包，读者可以通过使用官网（https://npmjs.org/）进行查询。下面演示基于Node.js开发的Web应用框架Express程序包的安装过程。

（1）打开Windows命令提示工具，执行如下代码：

```
npm search express
```

由于是首次进行搜索，所以NPM会先提示下载官网上注册程序包的索引集合，如图4.14所示。

```
npm
Microsoft Windows [版本 5.2.3790]
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>npm search express
npm WARN Building the local index for the first time, please be patient
npm http GET https://registry.npmjs.org/-/all
```

图4.14 NPM搜索提示

待下载完毕后，会出现一堆与Express程序包相关的项目提示，如图4.15所示。

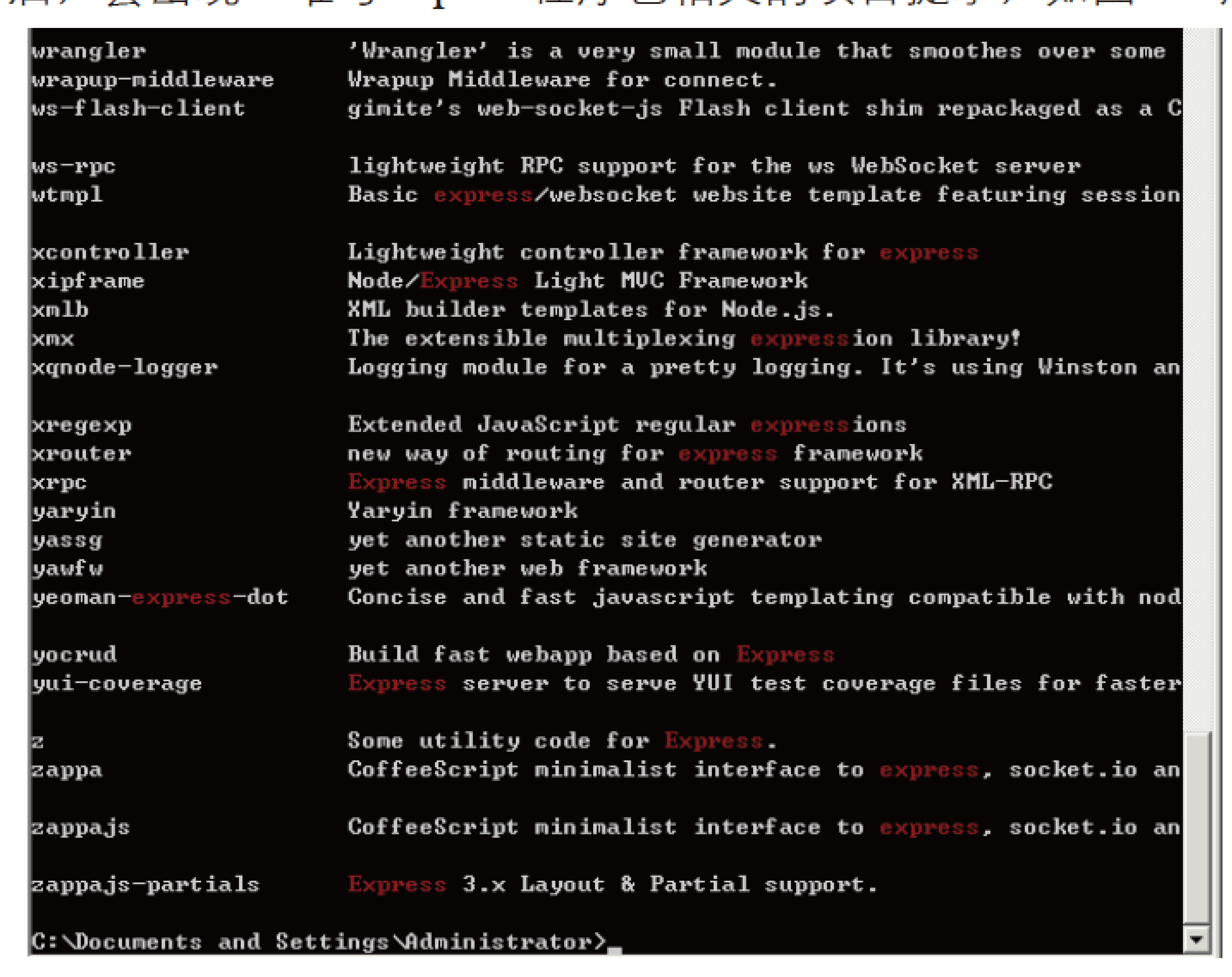


图4.15 搜索Express提示

（2）进行Express模块安装，执行如下代码：

```
npm install express
```

NPM开始从远程服务器下载Express相关安装代码，如图4.16所示。

```
C:\Documents and Settings\Administrator>npm install express
npm http GET https://registry.npmjs.org/express
npm http 200 https://registry.npmjs.org/express
npm http GET https://registry.npmjs.org/express/-/express-3.3.4.tgz
npm http 200 https://registry.npmjs.org/express/-/express-3.3.4.tgz
npm http GET https://registry.npmjs.org/connect/2.8.4
npm http GET https://registry.npmjs.org/commander/1.2.0
npm http GET https://registry.npmjs.org/range-parser/0.0.4
npm http GET https://registry.npmjs.org/buffer-crc32/0.2.1
npm http GET https://registry.npmjs.org/mkdirp/0.3.5
npm http GET https://registry.npmjs.org/methods/0.0.1
npm http GET https://registry.npmjs.org/send/0.1.3
npm http GET https://registry.npmjs.org/cookie/0.1.0
npm http GET https://registry.npmjs.org/fresh/0.1.0
npm http GET https://registry.npmjs.org/debug
npm http GET https://registry.npmjs.org/cookie-signature/1.0.1
npm http 200 https://registry.npmjs.org/commander/1.2.0
npm http GET https://registry.npmjs.org/commander/-/commander-1.2.0.tgz
npm http 200 https://registry.npmjs.org/mkdirp/0.3.5
npm http GET https://registry.npmjs.org/mkdirp/-/mkdirp-0.3.5.tgz
npm http 200 https://registry.npmjs.org/connect/2.8.4
npm http GET https://registry.npmjs.org/connect/-/connect-2.8.4.tgz
npm http 200 https://registry.npmjs.org/buffer-crc32/0.2.1
npm http 200 https://registry.npmjs.org/methods/0.0.1
npm http GET https://registry.npmjs.org/buffer-crc32/-/buffer-crc32-0.2.1.tgz
npm http GET https://registry.npmjs.org/methods/-/methods-0.0.1.tgz
```

图4.16 NPM开始从远程服务器下载Express相关安装代码

（3）下载结束后，“C:\Documents and Settings\Administrator”文件夹会多出一个名为“node_modules”的文件夹，里面存放了刚才下载的express和其相关的依赖程序包。

提示

Express是一个简洁而灵活的Node.js下Web应用框架，提供一系列强大特性帮助创建各种Web应用，丰富的HTTP工具以及来自Connect框架的中间件随取随用，创建强健、友好的API变得快速又简单。Express不对Node.js已有的特性进行二次抽象，只是在其之上扩展了Web应用所需的功能，官网地址为http://expressjs.com/。

之前已经介绍过，NPM除了可以下载使用他人远程共享的程序包，同时也可以发布自己的程序包，下面将演示如何发布自己的程序包。

（1）前往官网https://npmjs.org/signup注册一个账号。

（2）在本地计算机分区，此处为D盘上新建文件夹命名为npw，用于发布测试，在Windows命令行工具执行npm init代码创建程序包，创建期间NPM会出现多次提示输入信息，为了便于测试，这里只填写了第一项name信息为“html5-test”，表示该程序包的发布名称，如图4.17所示。



请注意NPM的程序包名称具有唯一性，读者必须使用没有被占用的名称，本例测试名为“html5-test”。

```
D:\npm>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sane defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (npm) html5-test
version: (0.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (BSD)
About to write to D:\npm\package.json:

{
  "name": "html5-test",
  "version": "0.0.0",
  "description": "ERROR: No README.md file found!",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": "",
  "author": "",
  "license": "BSD"
}


Is this ok? (yes) yes
npm WARN package.json html5-test@0.0.0 No README.md file found!
```

图4.17 创建属于自己的程序包

（3）程序包创建成功以后，执行npm adduser命令添加发布用户，按提示输入之前注册的账号，如图4.18所示。

```
D:\npm>npm adduser
Username: zhouyao
Password:
Email: yao.zhou@dianping.com
npm http PUT https://registry.npmjs.org/-/user/org.couchdb.user:zhouyao
npm http 409 https://registry.npmjs.org/-/user/org.couchdb.user:zhouyao
npm http GET https://registry.npmjs.org/-/user/org.couchdb.user:zhouyao
npm http 200 https://registry.npmjs.org/-/user/org.couchdb.user:zhouyao
npm http PUT https://registry.npmjs.org/-/user/org.couchdb.user:zhouyao/-rev/9-5
d0db93906e724e347b0f809a000d49d
npm http 201 https://registry.npmjs.org/-/user/org.couchdb.user:zhouyao/-rev/9-5
d0db93906e724e347b0f809a000d49d
```

图4.18 添加发布用户

（4）执行npm publish命令进行远程发布，如图4.19所示。

```
D:\npm>npm publish
npm WARN package.json html5-test@0.0.1 No README.md file found!
npm http PUT https://registry.npmjs.org/html5-test
npm http 409 https://registry.npmjs.org/html5-test
npm http GET https://registry.npmjs.org/html5-test
npm http 200 https://registry.npmjs.org/html5-test
npm http PUT https://registry.npmjs.org/html5-test/0.0.1/-tag/latest
npm http 201 https://registry.npmjs.org/html5-test/0.0.1/-tag/latest
npm http GET https://registry.npmjs.org/html5-test
npm http 200 https://registry.npmjs.org/html5-test
npm http PUT https://registry.npmjs.org/html5-test/-/html5-test-0.0.1.tgz/-rev/2
-08b2090de2e44870fa83d36f86377027
npm http 201 https://registry.npmjs.org/html5-test/-/html5-test-0.0.1.tgz/-rev/2
-08b2090de2e44870fa83d36f86377027
+ html5-test@0.0.1
```

图4.19 执行npm publish命令进行远程发布

（5）发布成功后，可以登录网站https://npmjs.org/搜索刚刚发布的“html5-test”程序包，或者通过命令“npm search html5-test”。

此外，用户还可以进行项目远端删除，命令如下：

```
npm unpublish -force
```

4.2.4 如何在Node.js中调试

调试代码是软件开发中必不可少的组成部分，快速地调试可以帮助开发人员迅速地查找和解决问题。本节将向读者介绍Node.js中的两种调试方法：内置调试器和基于Chrome浏览器调试。

1. 内置调试器

内置调试有点类似于网页开发中向代码添加关键词debugger的方法，比如在Chrome浏览器开发者工具中的JavaScript控制台输入如下代码：

```
var a=1;
debugger;
```

JavaScript控制台代码如图4.20所示。



图4.20 JavaScript控制台

按Enter键执行代码，Chrome进入调试模式，如图4.21所示。



图4.21 Chrome调试模式

Node.js的内置调试使用的也是类似的方法，下面介绍如何运行Node.js的内置调试。

（1）在文件夹“D:\npm”中新建名为“app.js”的文件，代码内容如下：

```
var a=1;
debugger;
```

（2）打开Windows命令行工具进入文件夹“D:\npm”，执行如下命令运行脚本：

```
node debug app.js
```

脚本代码执行成功后，效果如图4.22所示，表示进入调试状态。

（3）输入命令n，按Enter键执行下一步，此时脚本文件中的第一行代码被执行，如图4.23所示。

```
D:\npm>node debug app.js
< debugger listening on port 5858
connecting... ok
debug>
```

图4.22 脚本代码执行成功

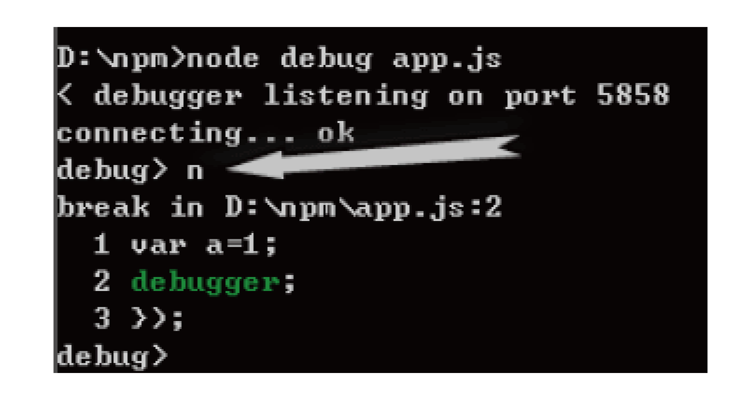


图4.23 输入字符n执行下一步

（4）输入命令repl，按Enter键进入脚本调试器的上下文交互式结果查看状态，如图4.24所示。

（5）输入a表示显示当前脚本上下文中的变量a的值，按Enter键执行，调试器给出a的值为1，如图4.25所示。

```
D:\npm>node debug app.js
< debugger listening on port 5858
connecting... ok
debug> n
break in D:\npm\app.js:2
  1 var a=1;
  2 debugger;
  3 });
debug> repl
Press Ctrl + C to leave debug repl
> _
```

图4.24 脚本调试上下文交互式结果查看状态

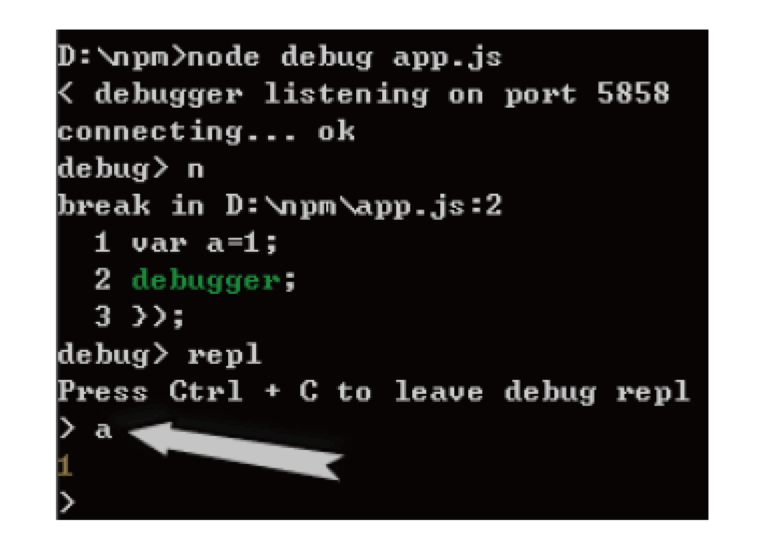


图4.25 输入并显示变量“a”的值

原生的内置调试器除了刚才使用的命令n、repl外，还提供了其他非常丰富的功能，命令列表使用说明如表4.1所示。

表4.1 内置调试命令列表

命令	说明
run	表示运行脚本（调试器启动自动运行），可缩写为r
cont	表示继续执行，直到遇到下一个断点，可缩写为c
next	表示单步执行，可缩写为n
step	表示单步进入函数，可缩写为s
out	表示单步跳出函数，可缩写为o
backtrace	表示打印当前执行帧的回溯，可缩写为bt
setBreakpoint	表示设置断点，默认为当前行，可缩写为sb
clearBreakpoint	表示移除断点，可缩写为cb
watch	表示添加表达式观察列表
unwatch	表示移除观察列表的表达式

（续表）

命令	说明
watchers	表示列出所有观察表达式和值（自动列在每个断点）
repl	表示进入脚本调试器的上下文交互式结果查看状态
restart	表示重新开始执行脚本
kill	表示结束脚本
list	表示列出脚本源代码某行的上下文（该行的前后）
scripts	表示列出所有加载脚本
breakOnException	表示断点于异常出现
breakpoints	表示列出所有断点信息
version	表示显示V8引擎版本

2. Chrome浏览器

另外一种调试Node.js的方法为Chrome浏览器调试，这里将用到程序包node-inspector，首先在Windows命令行工具中执行命令将程序包安装在全局环境下，命令如下：

```
npm install -g node-inspector
```

安装完毕后，继续执行node-inspector，启动该服务，命令如下：

```
node-inspector
```

提示

如果提示“warn – error raised: Error: listen EADDRINUSE”，表示默认的8080端口已经被占用，可以通过命令“node-inspector -web-port=新端口号”来指定一个未被占用的端口。

node-inspector启动成功后，会提示用户调试页面地址，如图4.26所示。

```
C:\Documents and Settings\Administrator>node-inspector
Node Inspector v0.3.1
   info  - socket.io started
Visit http://127.0.0.1:8080/debug?port=5858 to start debugging.
```

图4.26 node-inspector启动成功

此时node-inspector正在监听5858调试端口，打开Windows命令行工具进入目录“D:\npm”，继续使用之前的测试文件app.js，代码如下：

```
var a=1;
debugger;
```

执行命令如下：

```
node --debug-brk=5858 app.js
```

运行成功后提示debugger listening on port 5858。打开Chrome浏览器，进入“node-inspector”调试地址“http://127.0.0.1:8080/debug?port=5858”，页面如图4.27所示。

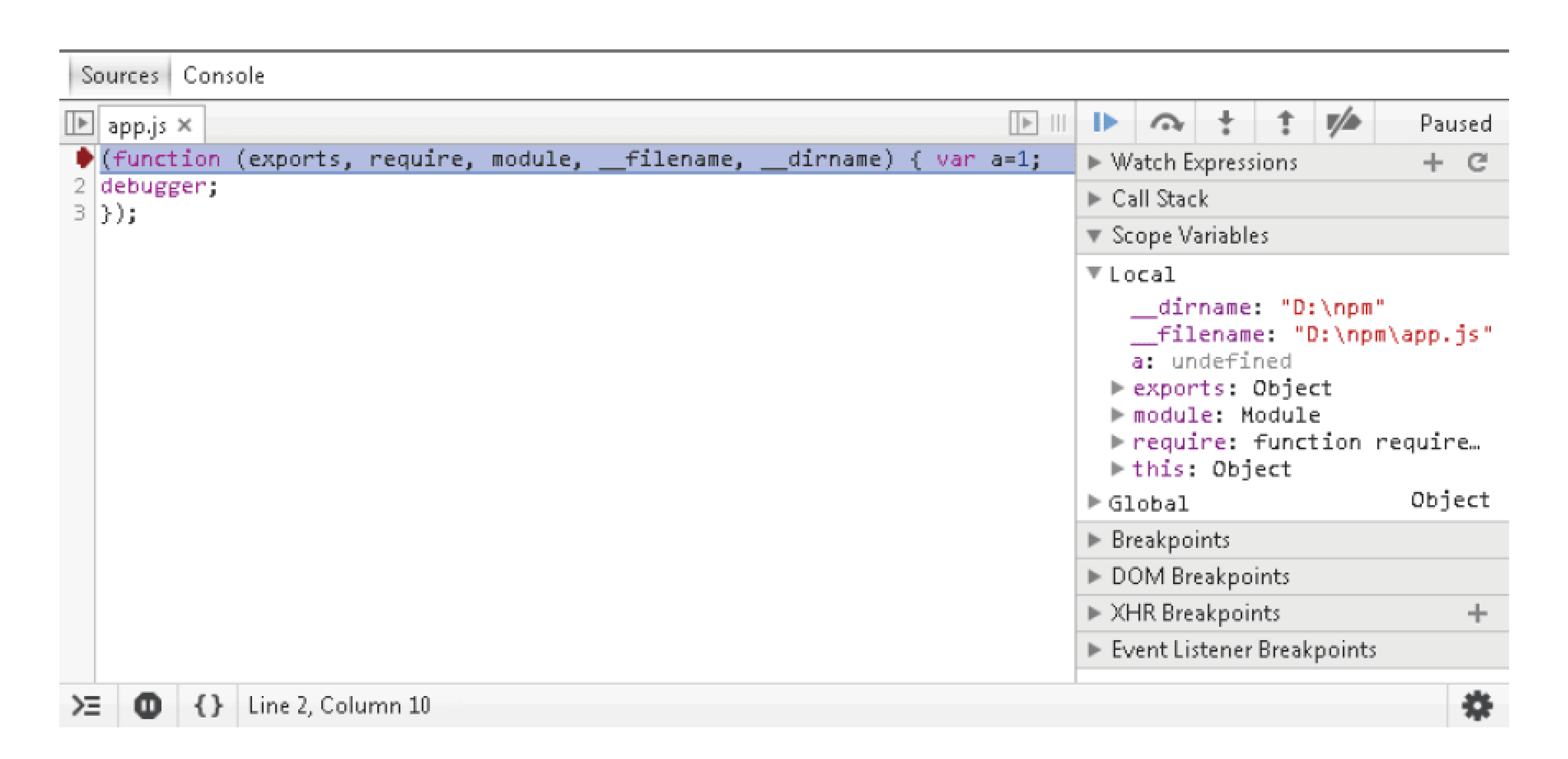


图4.27 node-inspector调试状态

相比较两种调试方法，笔者更偏向于使用node-inspector，该方法更接近于传统的浏览器调试模式。



node-inspector通过WebSocket将调试信息反馈给浏览器，用于打开的浏览器必须支持HTML 5的WebSocket功能，本节使用Chrome浏览器打开。

4.2.5 使用Node.js搭建Web Server

Node.js在其官网首页就已经说明其目的是构建快速、可伸缩的网络应用程序。下面就从最经典的Hello World示例程序开始，创建一个简单的Web服务器，示例代码如下：

```
var http = require('http');              // 引入http模块
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
      // 写入http状态和类型信息
  res.end('Hello World\n');              // 输出内容
}).listen(1337, '127.0.0.1');            // 创建服务，监听本地1337端口
console.log('Server running at http://127.0.0.1:1337/');
```

将Hello World代码放入新建的脚本文件example.js中，并执行命令如下：

```
node example.js
```

服务启动成功，Windows命令控制台提示如下：

```
Server running at http://127.0.0.1:1337/
```

打开浏览器，在地址栏输入地址http://127.0.0.1:1337/，按Enter键执行打开页面，效果如图4.28所示。

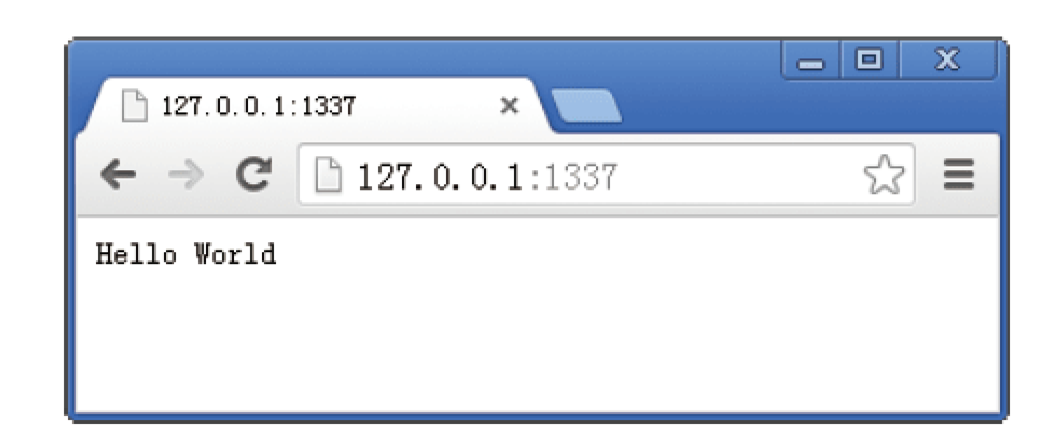


图4.28 Hello World页面

Hello World完成的工作虽然简单，但是Node.js凭借简单的几行代码完成一个服务器，足以看到Node.js作者在HTTP方面下的苦工。

下面继续给读者演示名为Express的基于Node.js的Web应用框架。先前的小节已经做过简单的介绍，Express提供一系列强大特性帮助创建各种Web应用，丰富的HTTP工具以及来自Connect框架的中间件随取随用，创建强健、友好的API变得快速又简单。

接着，通过一个示例向读者演示Express的使用，读者可以按照下列步骤进行操作，以Windows系统为例。

（1）在计算机上此处为D盘上新建文件夹，命名为express，用于存储模块和代码。

（2）打开Windows命令行工具，进入刚刚新建的express文件夹，执行命令如下：

```
cd d:express                                    // 转换路径
d:                                              // 进入express文件夹
node install express                            // 安装express程序包
```

等待express程序包下载完毕以后，express文件夹内多出一个新的名为“node_modules”的文件夹。

（3）进入“node_modules”文件夹内的“.bin”文件夹，里面包含两个文件，如图4.29所示。

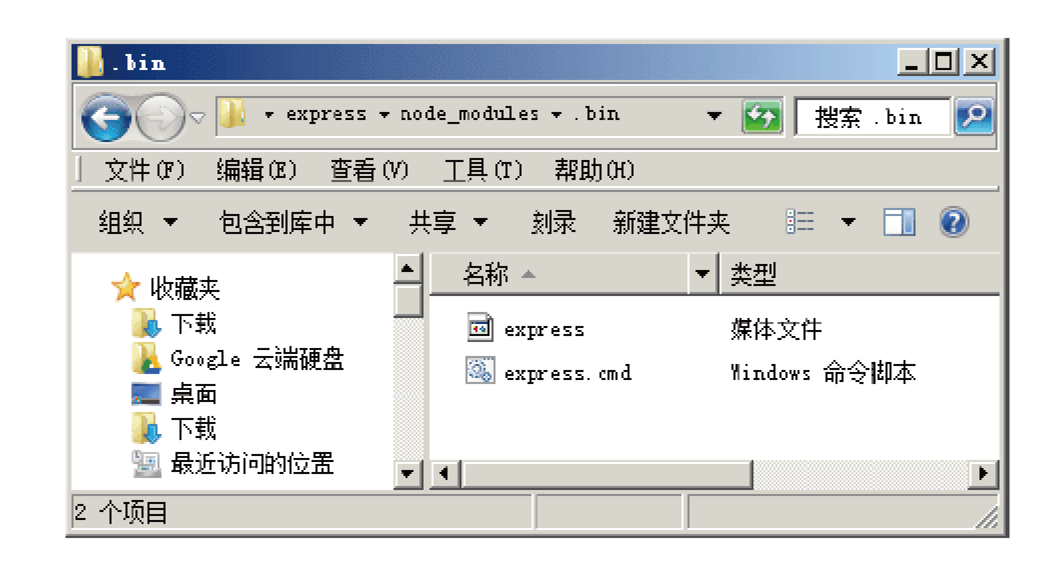


图4.29 “.bin”文件夹内容

（4）双击执行express.cmd命令脚本文件，出现Windows命令提示窗口，如图4.30所示。

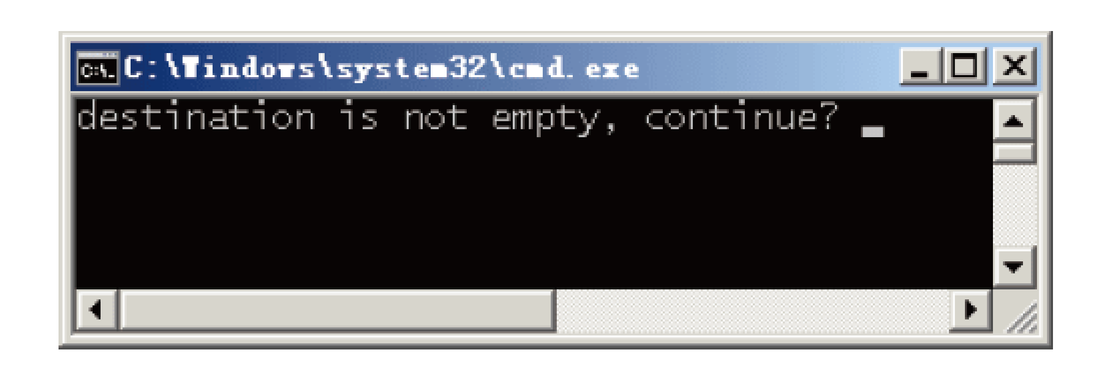


图4.30 双击执行express.cmd命令脚本文件

（5）输入命令yes，按Enter键执行，执行完毕后，会在“.bin”文件内生成Express的演示代码文件，如图4.31所示。

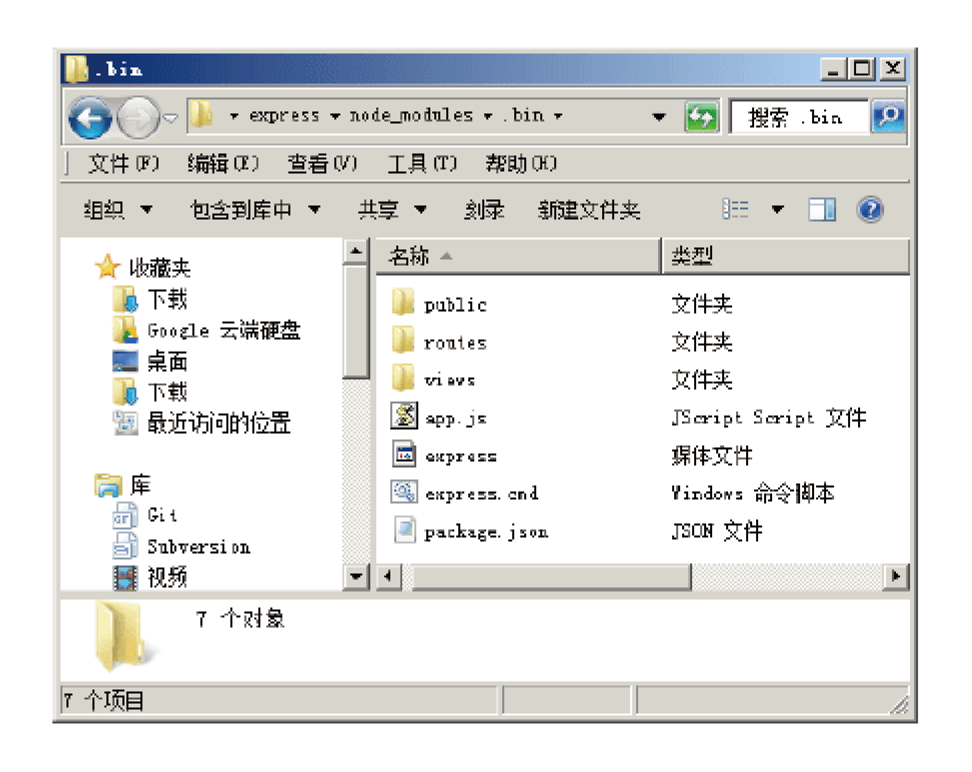


图4.31　演示代码文件生成完毕

生成示例文件说明如下。

- public：用于存放静态资源，如图片、样式表、脚本等。
- routes：存放对应路由规则的数据内容。
- views：存放页面模板。
- app.js：程序主执行文件。
- package.json：程序相关说明和依赖配置信息。

（6）打开Windows命令行工具，进入“.bin”文件夹，执行如下命令下载依赖程序包：

```
node install
```

等待下载依赖程序包Express和jade，还有对应的诸多依赖结束后，前期的准备工作完毕，接下来正式运行脚本启动基于Express框架的Web服务器。

（7）打开Windows命令行工具，进入.bin文件夹，执行如下命令启动服务：

```
node app.js
```

启动成功，命令行提示信息如下：

```
Express server listening on port 3000
```

（8）打开浏览器，在地址栏输入地址http://localhost:3000/，按Enter键执行打开页面，效果如图4.32所示。

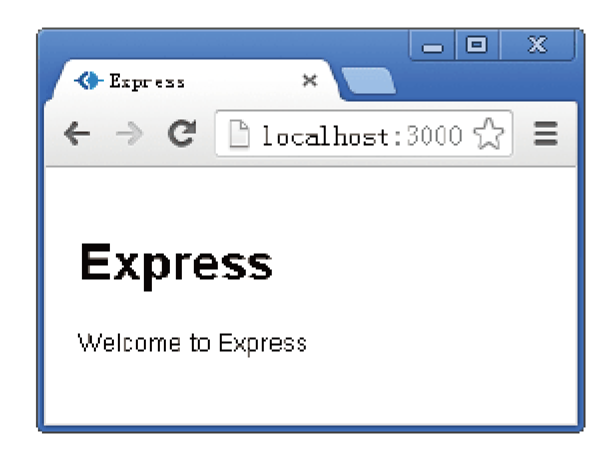


图4.32　Express框架示例页面



Express是一个非常强大的MVC框架，结合模板引擎jade在Web开发中游刃有余，读者想了解更该框架的使用，可以参考网站http://expressjs.com/内容。



4.3 jQuery框架

jQuery作为目前世界上最流行的JavaScript框架，也是本书示例的御用第三方框架，相信很多初次接触前端的读者对jQuery情有独钟。对于目前还不了解或者没有使用jQuery的读者，那么，请结束这种状态，迅速将jQuery掌握起来。

4.3.1 jQuery框架简介

jQuery由美国人John Resig于2006年1月14号在BarCamp NYC（http://barcamp.org）发布第一个版本，截止目前，全球最流行的网站中有超过59%的在使用该类库，在国内这一比例还更加惊人。jQuery优秀的应用程序接口设计使得操作非常容易。

jQuery有以下特色：

- 体积轻。
- 链式语法。
- 插件扩展。
- CSS选择器，支持从1~3。
- 文档丰富、易上手。

所以，不论使用的人是初学者还是资深专家，jQuery都将是首选。

4.3.2 jQuery常用API

API全称Application Programming Interface，中文意思为应用程序编程接口，API是对所需知识的抽象，将系统的复杂性隐藏起来，并向外部提供易理解、一致、可预见、简洁的调用方式。jQuery在API的设计上是相当出色的，按照最新的官网提供的1.9版本API文档，可将jQuery提供的API分为Ajax、属性、回调对象、核心、DOM操作、CSS、数据、延迟对象、尺寸、效果、事件、表单、内部构建、操作、位置、选择器、遍历，不过在日常的开发中，并非所有分类都会使用到。下面讲解部分被频繁使用的API。

1. 选择器

jQuery选择器提供了快速定位元素的方法，其继承了CSS和Path语言的部分语法，允许通过标签名、属性名称、内容对元素进行精准的选择，用户在使用时可以完全不用考虑浏览器的兼容问题。学习使用选择器是学习jQuery的第一步，表4.2整理了最常用的选择器API。

表4.2 选择器常用API

使用	说明
$('#id')	选取一个页面元素ID
$('.class')	选取所有样式类包含class的元素
$('tagname')	选取所有标签名为tagname的元素
$("[attribute|='value']")	attribute表示一个属性名，value表示属性值
$(":checked")	表示选框被选中，适用于复选框和单选项
$(selector).find(selector)	获取元素下的所有筛选元素
$(selector).closest(selector)	从当前元素开始向上遍历，直到找到选择器的一个匹配为止

2．事件

浏览器事件指的是用户打开页面到关闭页面期间浏览器的动作和对用户操作的响应。如单击页面上的按钮、鼠标的移动、浏览器大小的调整等。jQuery文档在事件区域有7大块：浏览器事件、文档加载、事件绑定、事件对象、表单事件、键盘事件、鼠标事件。表4.3整理了jQuery事件中最常用的部分。

表4.3 事件常用API

使用	说明
$(selector).bind(event,data,function)	添加一个或多个事件处理程序
$(selector).unbind(event,function)	删除一个或多个事件处理程序
$(selector).trigger(event,[param1,param2,...])	触发元素的指定事件类型
$(document).ready(handler)	当DOM准备就绪时，执行的一个函数

3．属性

属性分为两类：元素属性和对象属性。所谓的元素属性指的就是附加在元素上的信息，英文称为attribute，代码如下所示：

```
<div class="header" style="color:red" data-content="xxx"></div>
```



其中的class、style、data-content即元素属性。

另外一种是对象属性，英文称为property，指的是对象本身具有的特性。世间的万物都可以看作是一个对象，如果将一本书看作为一个对象，那么书本的颜色、大小、页数、名称、作者、形状就是书这个对象的属性，代码如下所示：

```
var book = {                                          // 书对象
    color : 'blue',                                   // 蓝颜色
    size : '900*600',                                 // 900×600大小
    page : 500,                                       // 500页
    name : 'html5',                                   // 书名为html5
    author : 'yao.zhou',                              // 作者yao.zhou
    shape : ''rectangle'                              // 书的形状为矩形
}
```

属性的设置是JavaScript开发中最频繁的操作，表4.4整理了jQuery属性操作中最常用的部分。

表4.4 属性操作常用API

使用	说明
$(selector).addClass(class)	元素添加样式类
$(selector).removeClass(class)	移除元素样式类
$(selector).attr(attribute)	获取元素的属性值
$(selector).attr(attribute,value)	设置元素的属性值
$(selector).val()	获取元素value 属性的值
$(selector).val(value)	设置元素value 属性的值
$(selector).html()	获取元素的内容（innerHTML）
$(selector).html(content)	设置元素的内容

4. DOM操作

DOM是Document Object Model的简写，既文档对象模型，由一系列对象组成，是访问、检索、修改HTML文档内容与结构的标准方法，可以将其理解为以面向对象方式描述的文档模型，使用者可以添加、移除、改变、排列页面的结构，表4.5整理了jQuery中DOM操作最常用的部分。

表4.5 DOM操作常用API

使用	说明
$(selector).append(content)	将元素或者html字符串插入获取元素内部的尾部
$(selector).prepend(content)	将内容插入获取元素同级的后面
$(selector).after(content)	将元素或者html字符串插入获取元素内部的顶部
$(selector).before(content)	将内容插入获取元素同级的前面
$(selector).remove()	从DOM中去掉所有匹配的元素

5. CSS

CSS全称为Cascading Style Sheet，即级联样式表，通过设置样式表，可以控制HTML中各元素的显示属性，如字体颜色、大小、背景、粗细、显示方式等，表4.6整理了jQuery中CSS操作最常用的部分。

表4.6 CSS操作常用API

使用	说明
$(selector).css(name)	获取元素的CSS属性值
$(selector).css(name,value)	设置元素的CSS属性值
$(selector).offset()	获取元素相对于文档的位置
$(selector).offset(value)	设置元素相对于文档的位置
$(selector).position()	获取元素相对于父元素的位置

6. Ajax

全称Asynchronous JavaScript and XML，是一种网页开发技术，核心是JavaScript对象XmlHttpRequest，XmlHttpRequest允许用户向服务器提交请求并获取信息，而不用重新刷新页面。jQuery封装了Ajax并提供了一堆非常友善的API，表4.7整理了jQuery中Ajax操作最常用的部分。

表4.7 Ajax操作常用API

使用	说明
jQuery.get()	使用一个HTTP GET请求从服务器加载数据
jQuery.post()	通过服务器HTTP POST请求加载数据
jQuery.getJSON()	使用一个HTTP GET请求从服务器加载JSON编码数据
jQuery.ajax()	执行一个异步的HTTP（Ajax）的请求

通上以上的API基本能完成一般网站的前端开发工作，读者也可以前往jQuery的官方文档http://api.jQuery.com/学习更多相关知识。



4.4 其他实战开发技巧

一谈到前端的实战，读者可能首先会想到兼容各种浏览器的奇异技巧，想到为了测试前端兼容问题，需要在电脑上安装IE、Chrome、Firefox、Opera，甚至为了测试纯正的IE 6还需要安装一台虚拟机。是的，没有错，如果励志要做一位合格的前端工程师，那么这些一个都不能少。本节介绍的开发技巧不会过多的纠缠在兼容性上，更多的是如何提高前端调试效率，学习下面的技巧并融会贯通，可以帮助开发者更快地定位线上问题，迅速地判断并做出修复。

4.4.1 如何在Chrome浏览器调试脚本

每种浏览器都有自己的调试工具，如IE的开发人员工具、Firefox的Firebug，各种主流的浏览器调试工具都是大同小异，这里将要介绍的是Chrome浏览器的开发者工具，可以通过使用快捷键F12调出工具界面，如图4.33所示。

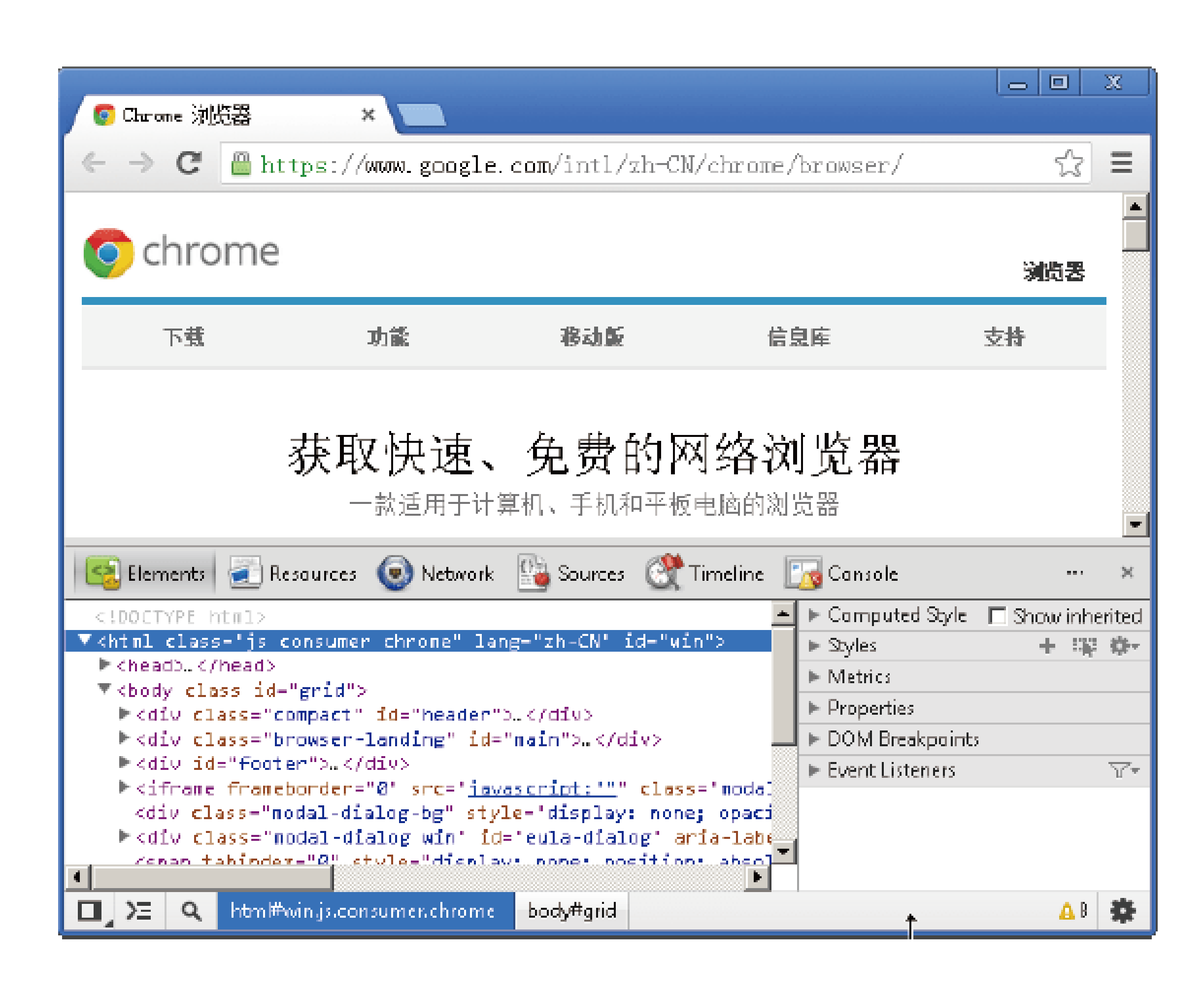


图4.33 Chrome浏览器开发者工具

工具条上面提供了6大默认功能Elements、Resources、Network、Sources、Timeline、Console。

（1）元素面板（Elements），如图4.33所示。右侧面板分别是Computed Style（计算所得样式）、Styles（页面样式）、Metrics（页面测距）、Properties（HTML属性）、DOM Breakpoints（DOM断点）、Event Listeners（事件监听器）。

在元素面板左侧HTML结构展示区，单击鼠标右键选中一个元素，出现元素修改菜单，如图4.34所示。

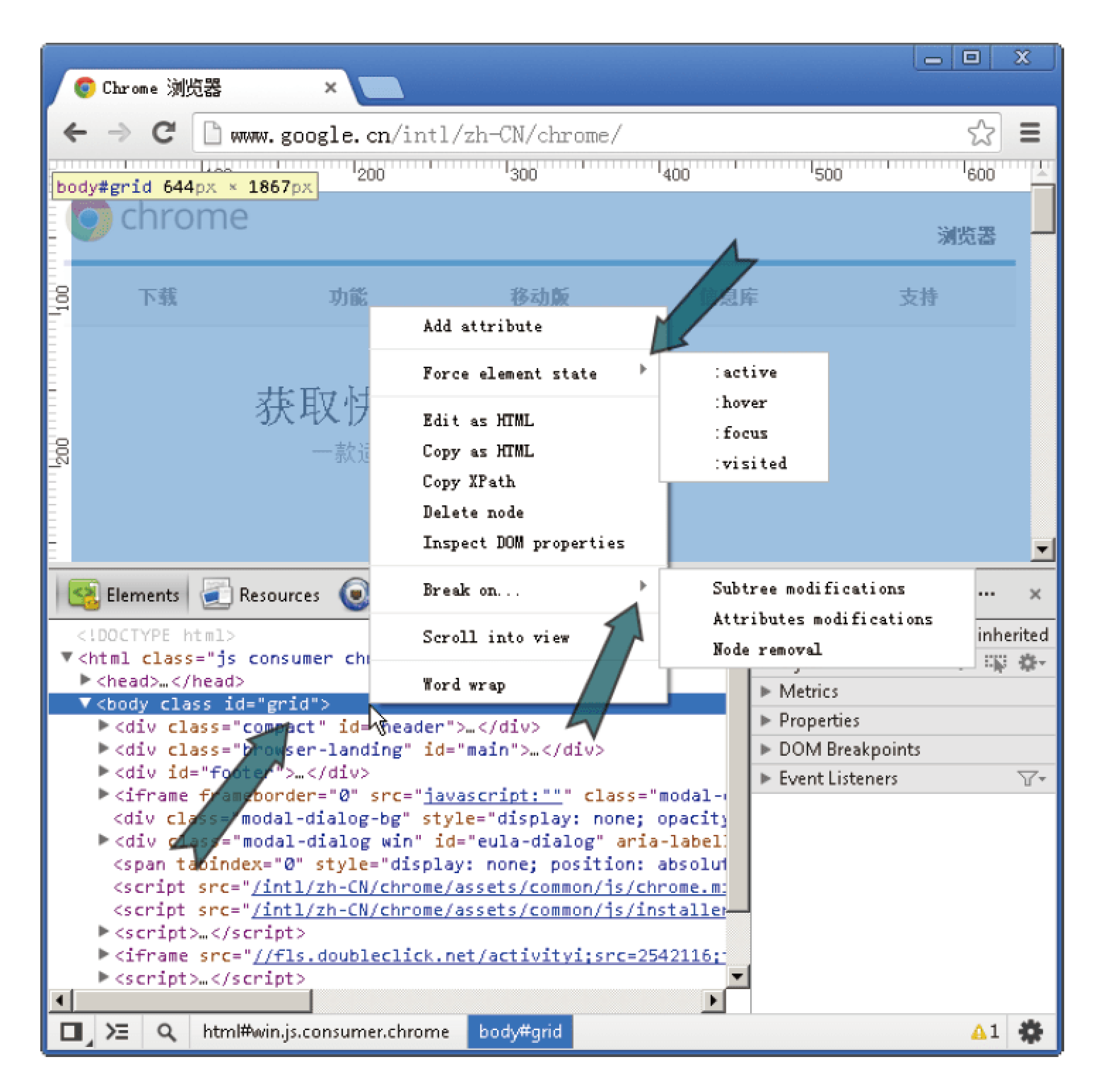


图4.34 单击鼠标右键选中一个元素

由上到下依次是：Add attribute（在元素上添加一个属性）、Force element state（强行设置元素状态）、Edit as HTML（以HTML方式编辑元素）、Copy as HTML（复制元素HTML结构内容）、Copy XPath（获取元素XPath）、Delete node（删除元素）、Inspect DOM properties（检查元素属性）、Break on（断点）、Scroll into view（视窗滚动到元素视线区）、Word wrap（自动换行）。

其中Force element state选项还有4种状态active（被按下时）、hover（悬浮时）、focus（焦点时）、visited（访问后）。Break on选项有三种状态Subtree modifications（子树发生修改时暂停进入断点）、Attributes modifications（元素属性发生修改时进入断点）、Node removal（元素被删除时进入断点）。

（2）资源面板（Resources），如图4.35所示。

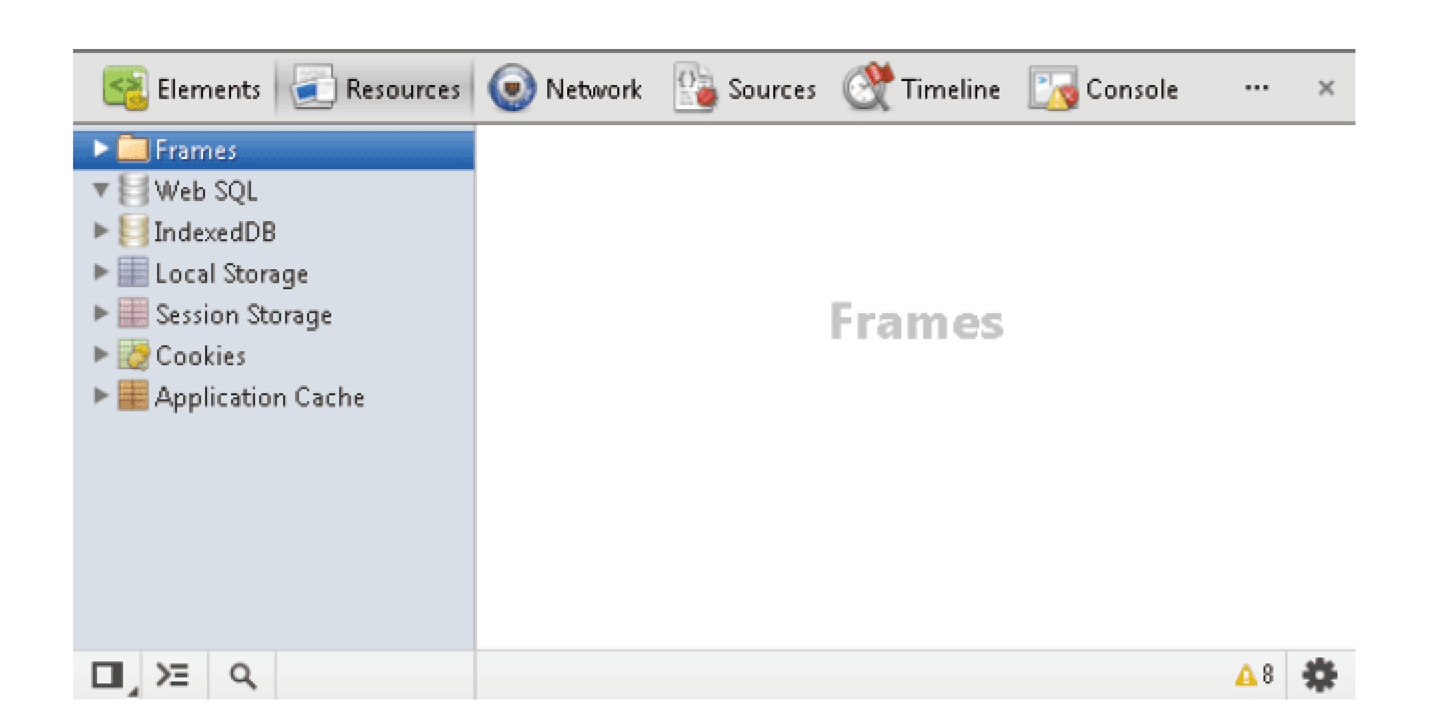


图4.35　资源面板

在资源面板的左侧可以依次看到Frames、Web SQL、IndexedDB、Local Storage、Session Storage、Cookies、Application Cache。Frames栏列出了页面上引入的所有窗口框架和其对应的资源，其余栏除了Cookies外都是HTML 5新增的功能，用户可以单击打开每项内容进行查看和管理，这里拿Local Storage举个例子，查看资源面板右侧的数据栏，在数据行上单击无鼠标右键，出现带有Edit和Delete功能的菜单栏，用户可以通过该菜单栏进行即时的数据编辑，如图4.36所示。

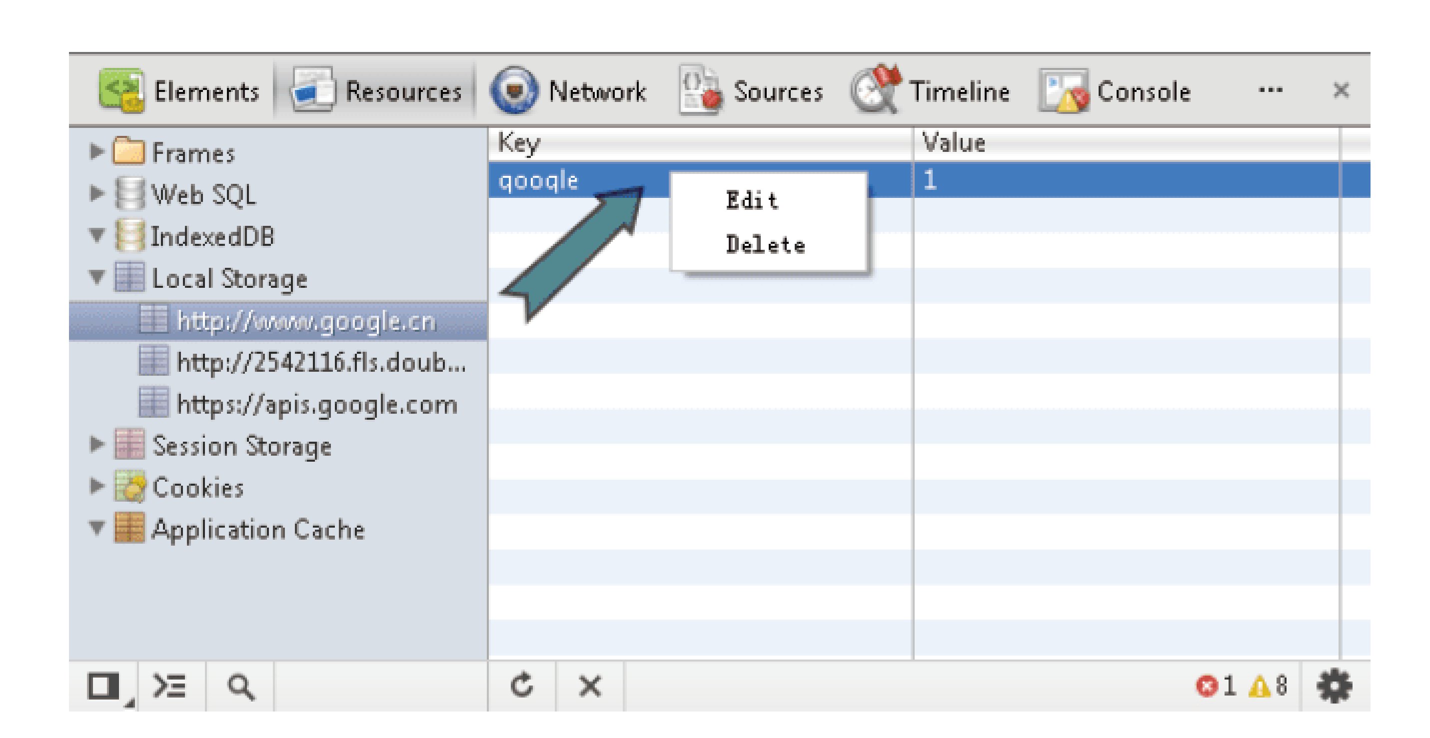


图4.36　Local Storage

（3）网络面板（Network），如图4.37所示为网址http://www.google.cn/intl/zh-CN/chrome/对应的网络瀑布图。

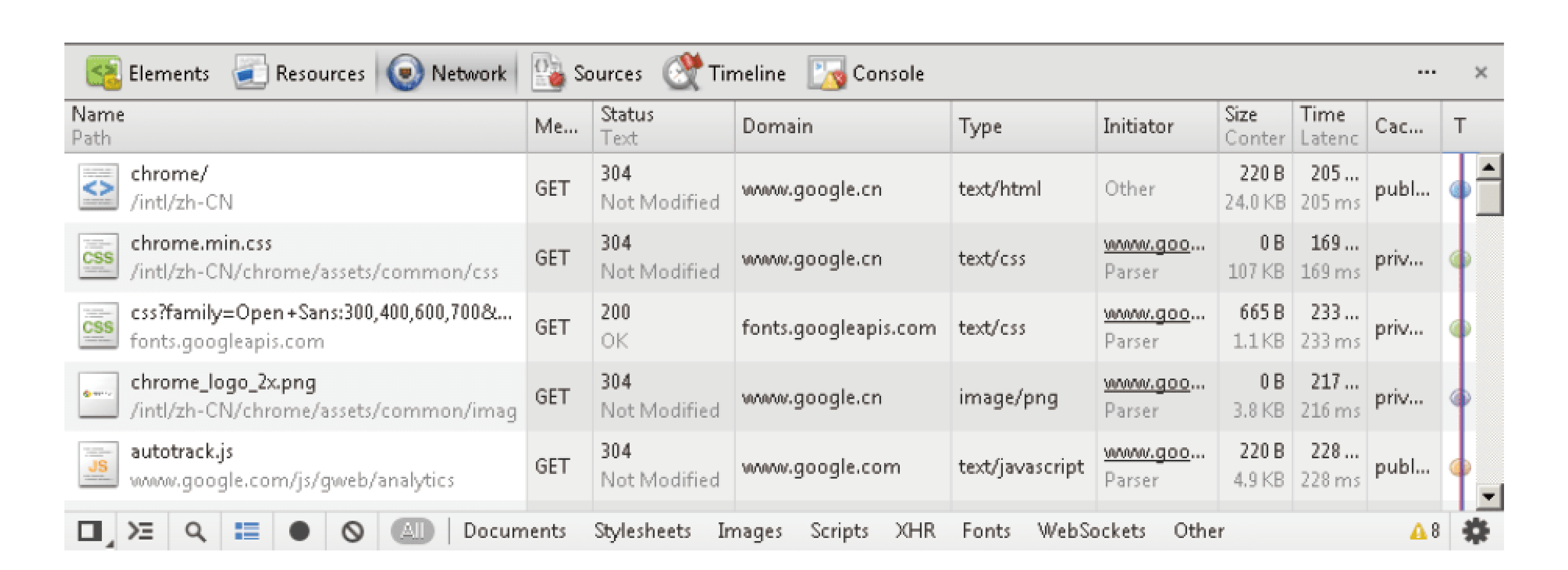


图4.37　网络瀑布题

通过瀑布图可以查看网络请求的具体细节，还可以通过面板底部的请求类型进行分类查看，如Documents、Stylesheets、Images、Scripts、XHR、Fonts、WebSockets、Other。请求的明细信息除了默认的一些外，还可以添加额外的信息，在面板请求标题处单击鼠标右键，出现菜单如图4.38所示。

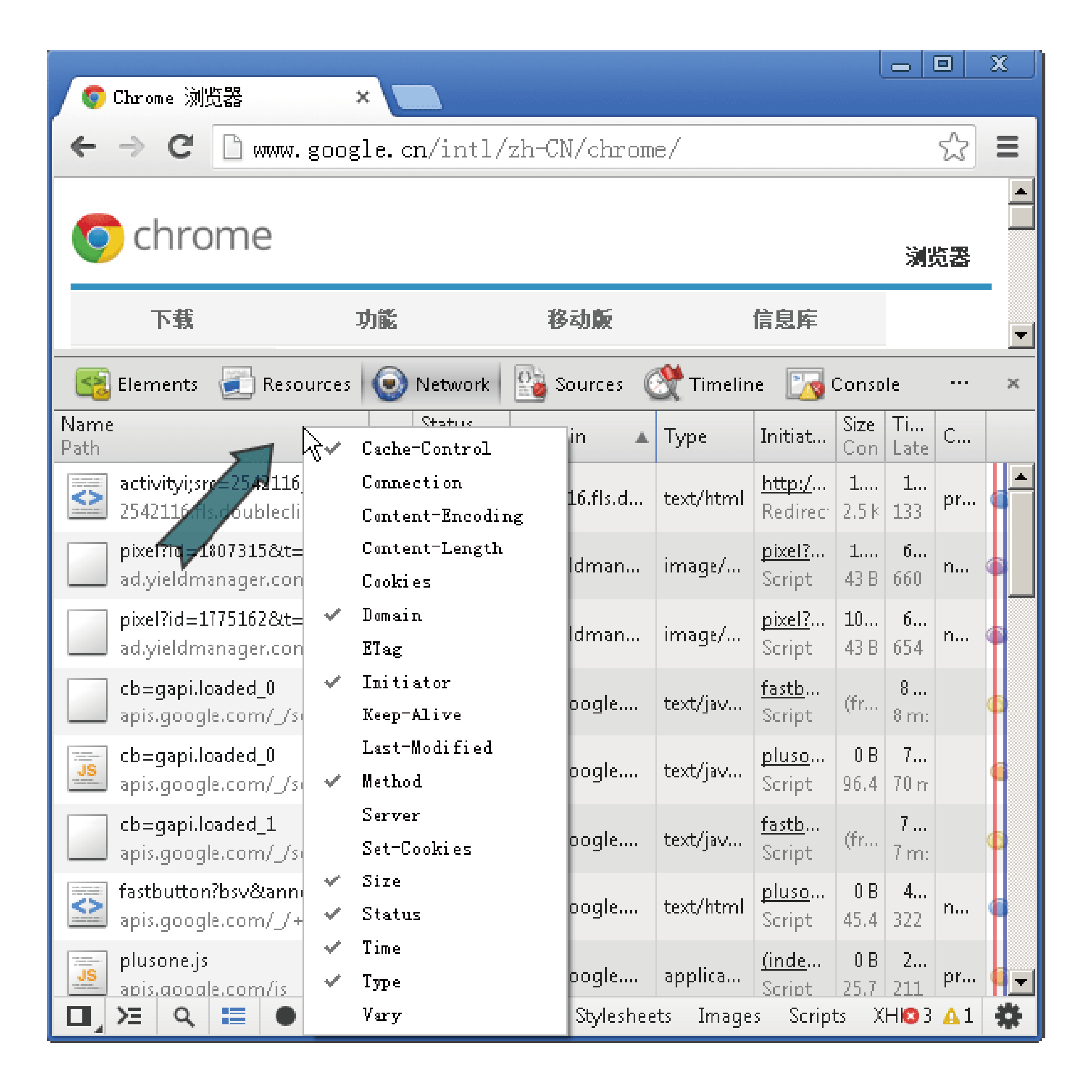


图4.38 在面板请求标题处单击鼠标右键

菜单选项的名称对应的说明如表4.8所示。

表4.8 菜单选项寿命

功能名	说明
Cache-Control	控制网页的缓存，常见的值有private、no-cache、max-age、must-revalidate等，默认为private
Connection	请求连接方式，如Keep-Alive、close等
Content-Encoding	采用的编码方式，如gzip
Content-Length	消息内容长度
Cookies	Cookie信息
Domain	域名信息
ETag	被请求变量的实体值
Initiator	请求发起者
Keep-Alive	保持连接特性
Last-Modified	最后资源修改时间
Method	请求类型，如GET、POST
Server	服务器类型
Set-Cookies	响应添加的cookie信息
Size	请求大小
Status	返回的HTTP状态码，如200、304等
Time	请求耗时
Type	请求类型，如image/gif、application/x-JavaScript等
Vary	中间缓存服务器的缓存依据

单击网络面板列表中的单条请求，面板右侧出现该条请求的详细信息，有Headers、Preview、Response、Cookies、Timing，如图4.39所示。

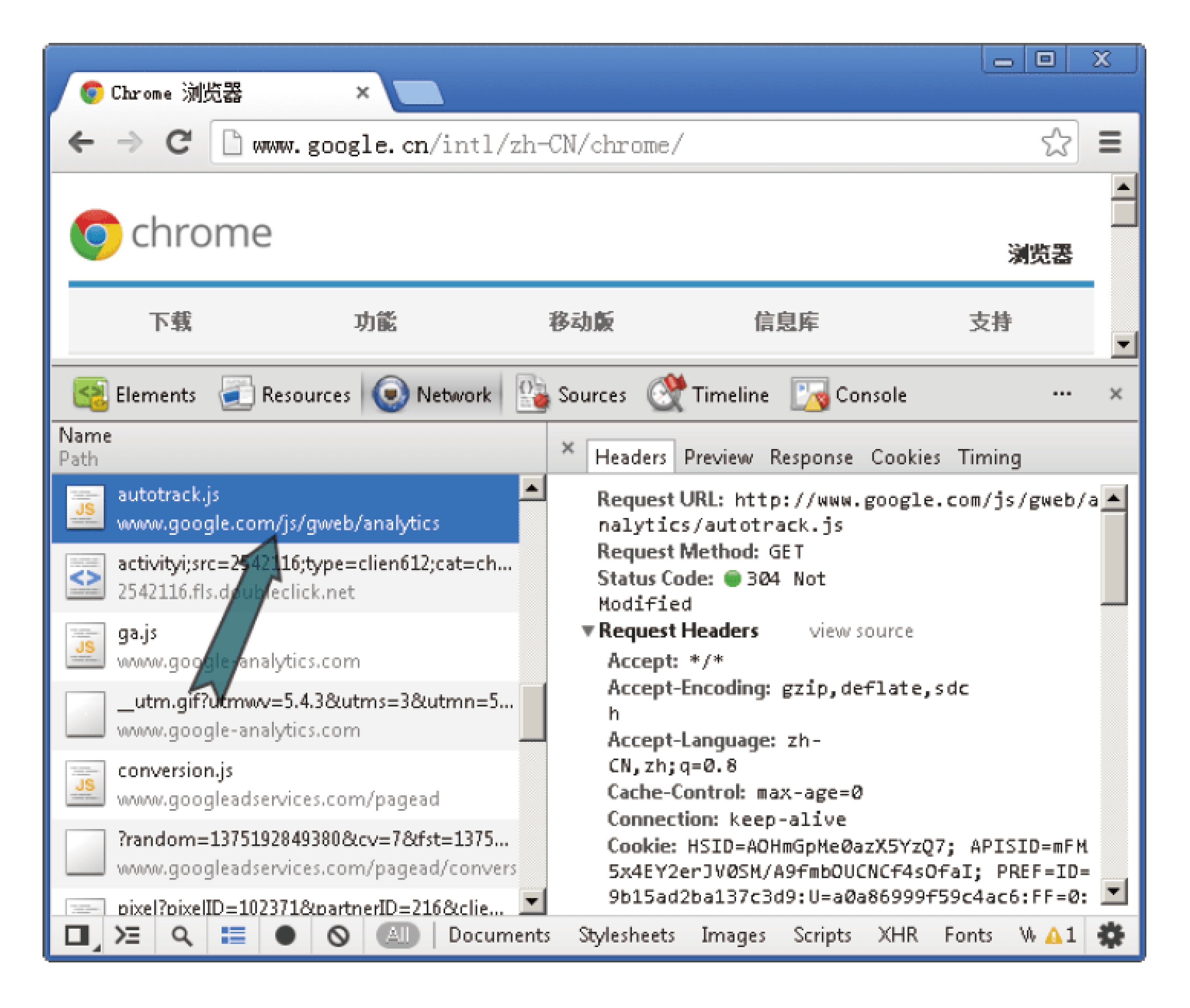


图4.39 请求详细信息

（4）源码面板（Sources），该区域功能主要用户查看和代码调试，如图4.40所示。

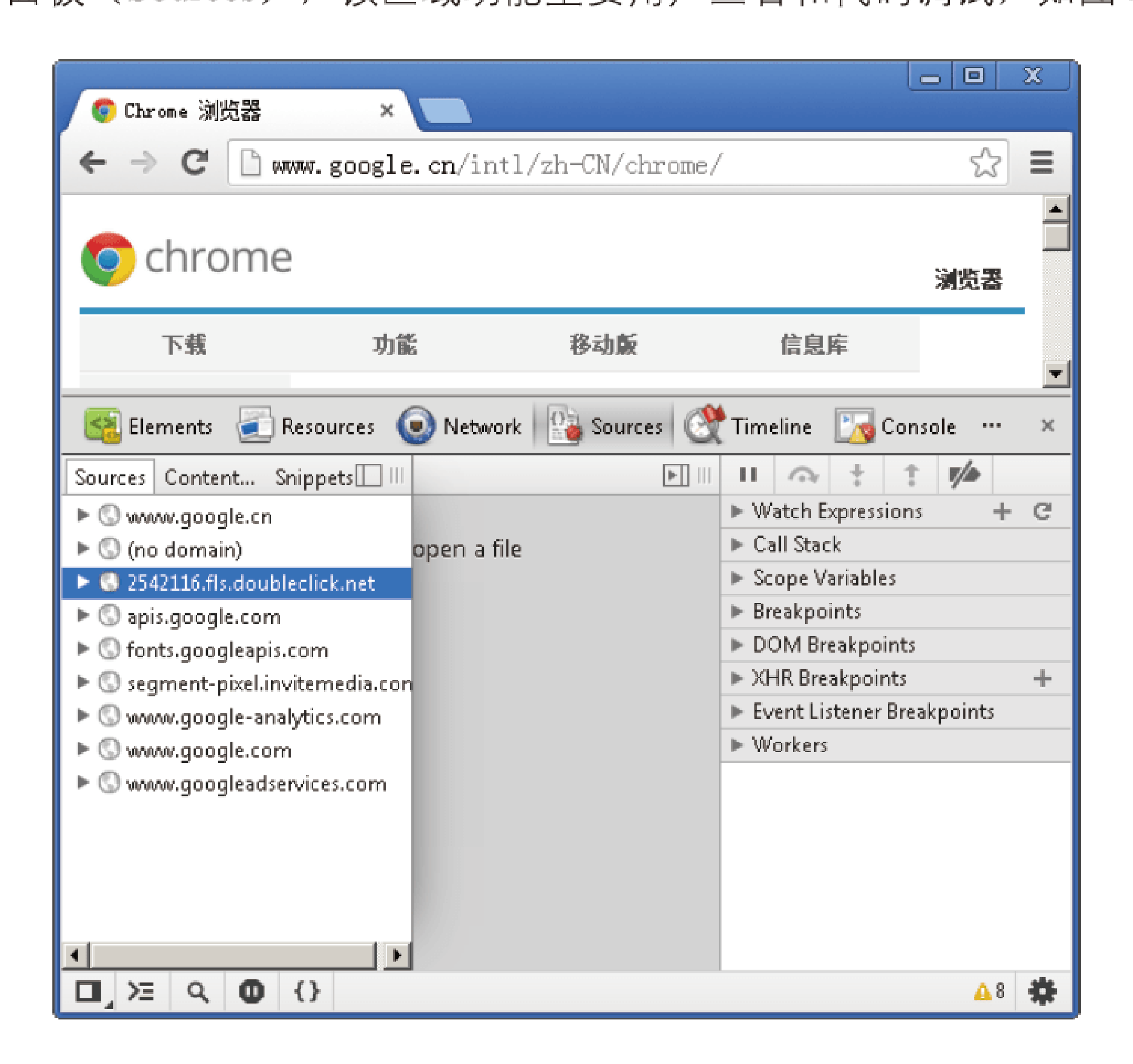


图4.40 源码面板

对于长期从事JavaScript的开发人员来说，使用最多的应该就是源码面板。面板的左侧是页面引用文件的完整列表，当页面引用的脚本文件非常多时，使用文件列表并不能很快的帮忙定位代码文件，这时候可以使用组合键Ctrl+O，出现如图4.41所示的文件查询面板。

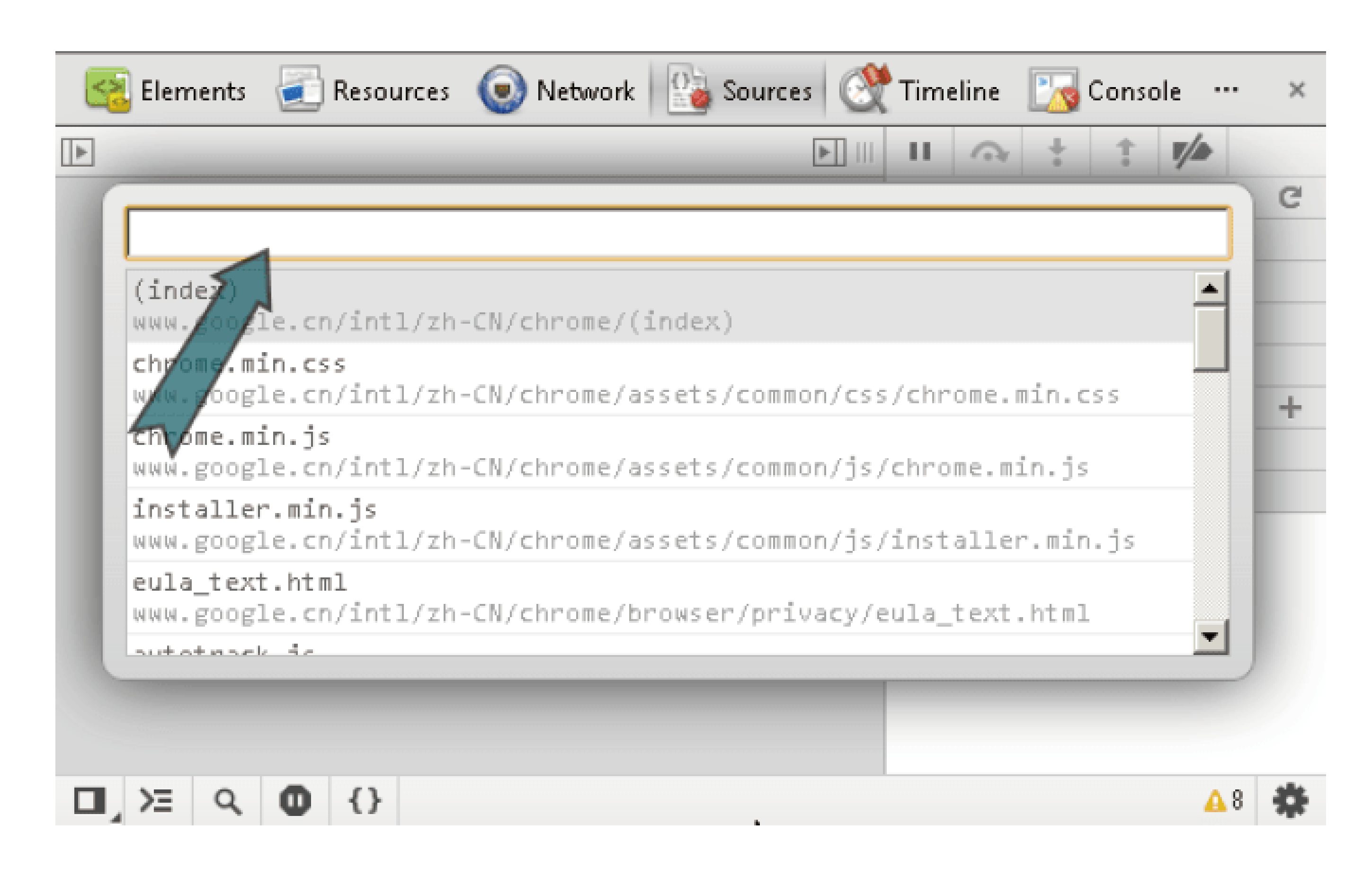


图4.41 文件查询面板

另外，如果想查询文件中的代码关键字，可以使用组合键Ctrl+Shift+F，在页面引用的脚本代码中查询带有google关键字的信息，如图4.42已经通过箭头标出查找后的高亮关键字信息。

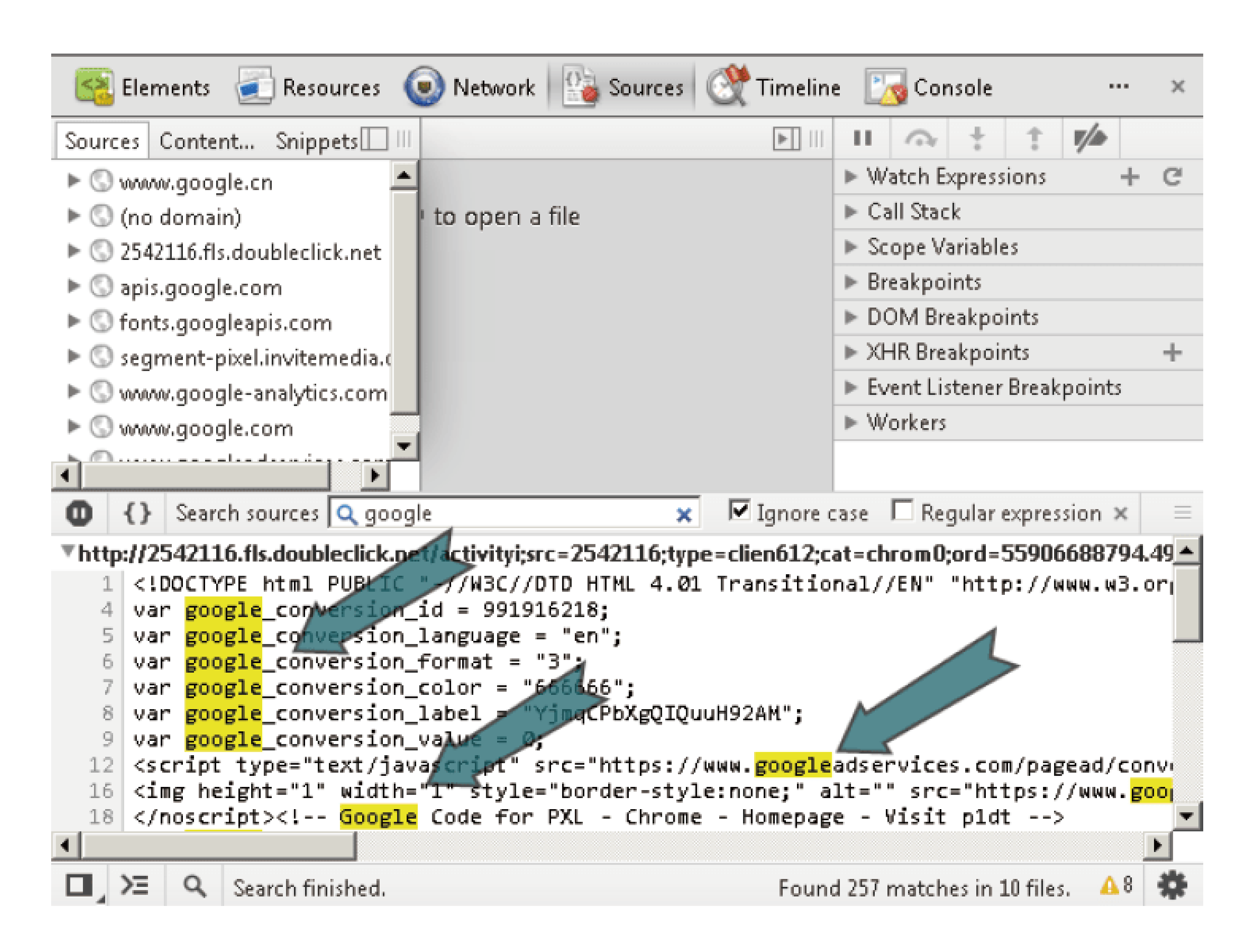


图4.42 使用代码查询面板查找代码中带有google关键字信息

（5）时间线面板（Timeline），可以检测Events（事件）、Frames（框架）、Memory（内存）三种类型，如图4.43所示。

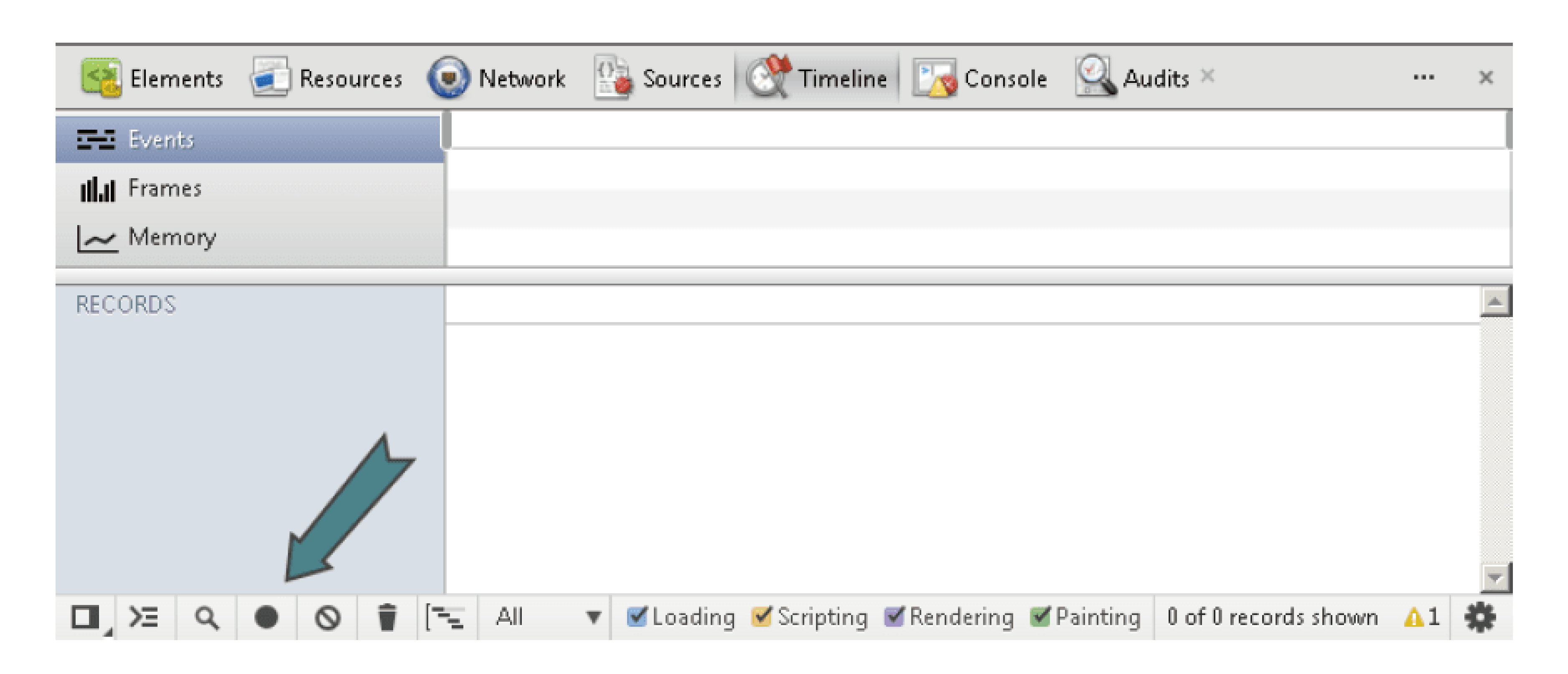


图4.43　时间线面板

如果使用者要进行记录，需要单击如图4.43箭头所示的圆点按钮，单击按钮变红后，面板就会显示每次执行操作的变化情况。该面板主要用于浏览器性能方面的调试，由于传统的网站正在向应用方向转型，页面功能日趋复杂，所以对于传统的网络优化是无法满足日益变化的复杂需求的，浏览器性能优化将是未来前端优化课题中不可或缺的一个重要部分。

（6）控制台（Console），这块也是日常前端经常使用的部分，用户可以在上面直接书写脚本代码，同时Chrome浏览器还非常友善地进行代码提示。

4.4.2 如何通过Fiddler加速开发

Fiddler是Windows底下最强大的请求代理调试工具，监控任何浏览器的HTTP/HTTPS流量，篡改客户端请求和服务器响应，解密HTTPS Web会话，图4.44为Fiddler原理示意图。

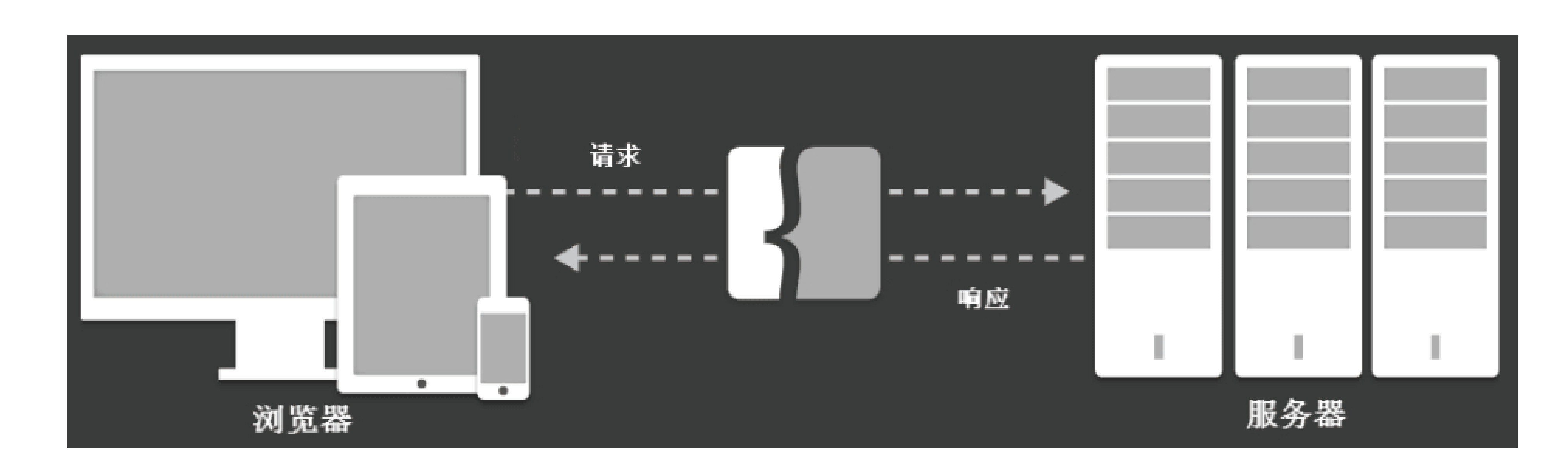


图4.44 Fiddler原理示意题

Fiddler安装的系统要求为Windows XP或Windows 8中的版本，其中Fiddler 2依赖于Microsoft.NET Framework 2.0，最新的Fiddler 4依赖于Microsoft.NET Framework 4.0。

Fiddler功能很多，在这里介绍一项最常用的代理功能。假使在维护的网站线上有个功能出现脚本问题，这时候采用传统的方法，将网页内容完整的保存到本地，然后调试对应的代码，很显然这种方法显得有点笨重，Fiddler解决这个问题显得游刃有余，其要做的是，将对应问题脚本保存到机器本地，修改脚本并通过Fiddler代理，下面通过一个百度首页的操作示例演示该过程。

（1）开启Fiddler，选中其右侧AutoResponder标签页，选中Enable automatic responses和

Unmatched requests passthrough复选框，如图4.45所示。

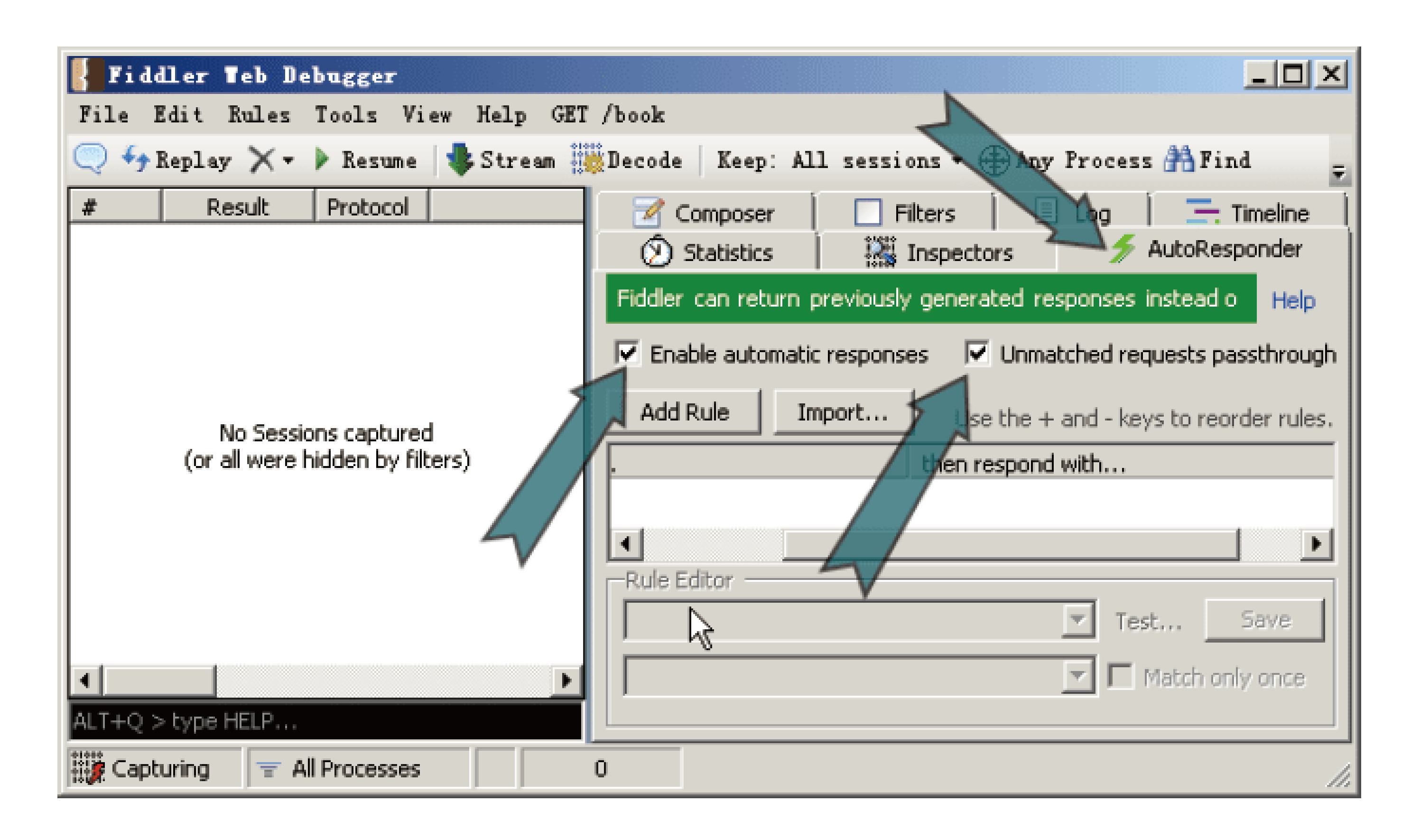


图4.45 开启Fiddler

（2）打开Chrome浏览器，在地址栏内输入http://www.baidu.com并按Enter键进入，此时百度首页的请求会被完整的显示在Fiddler左侧的列表中，如图4.46所示。

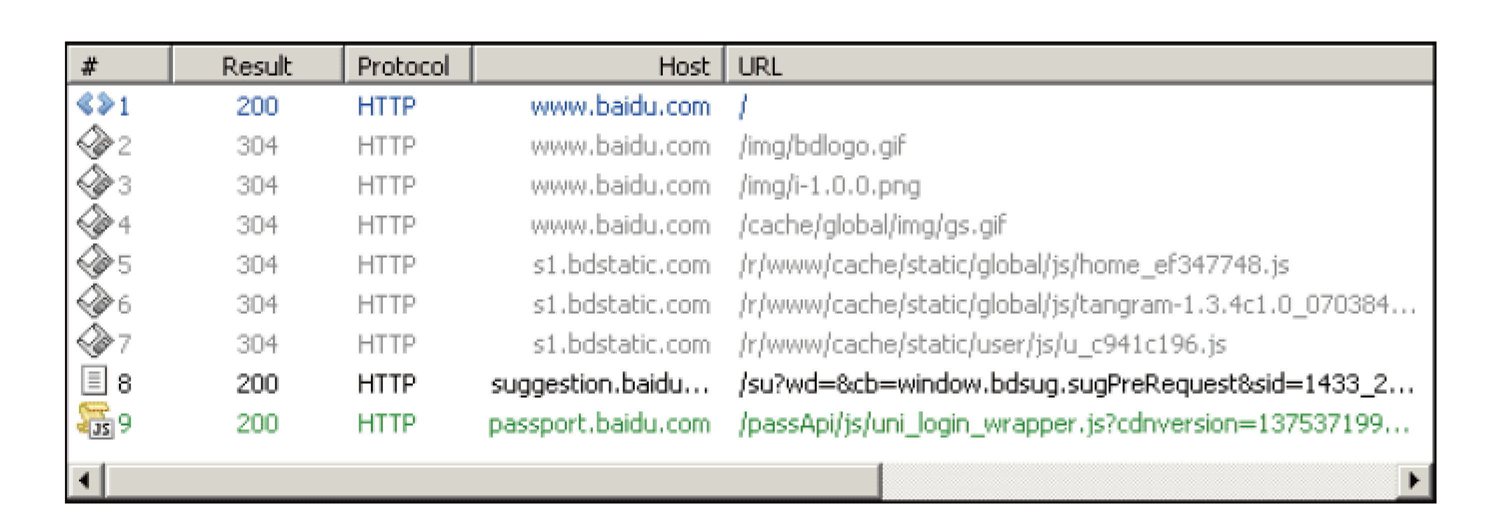


图4.46 百度首页请求列表

（3）选中列表中的第五条请求（该请求为JavaScript脚本），在该请求上方单击并拖动至右侧的AutoResponder标签页下方空白的列表中，如图4.47所示。

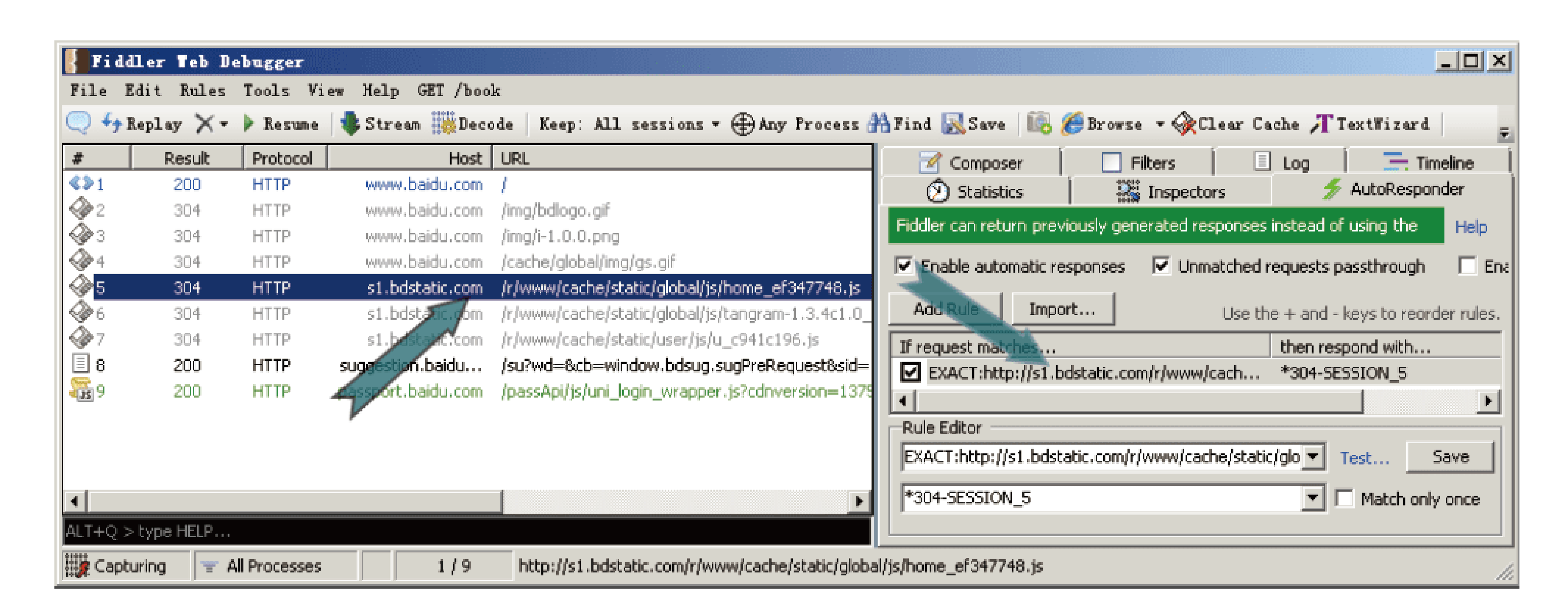


图4.47 代理一条脚本请求

（4）复制第五条请求的URL地址，使用浏览器打开并将脚本内容保存到机器本地文件夹

（5）在刚才保存的脚本末尾添加一行代码：

```
document.body.style.backgroundColor='black'        // 修改页面背景色为黑色
```

（6）修改Fiddler右侧AutoResponder标签页下方列表的“then respond width..”列，将其指向本地保存的脚本代码地址，如图4.48所示。

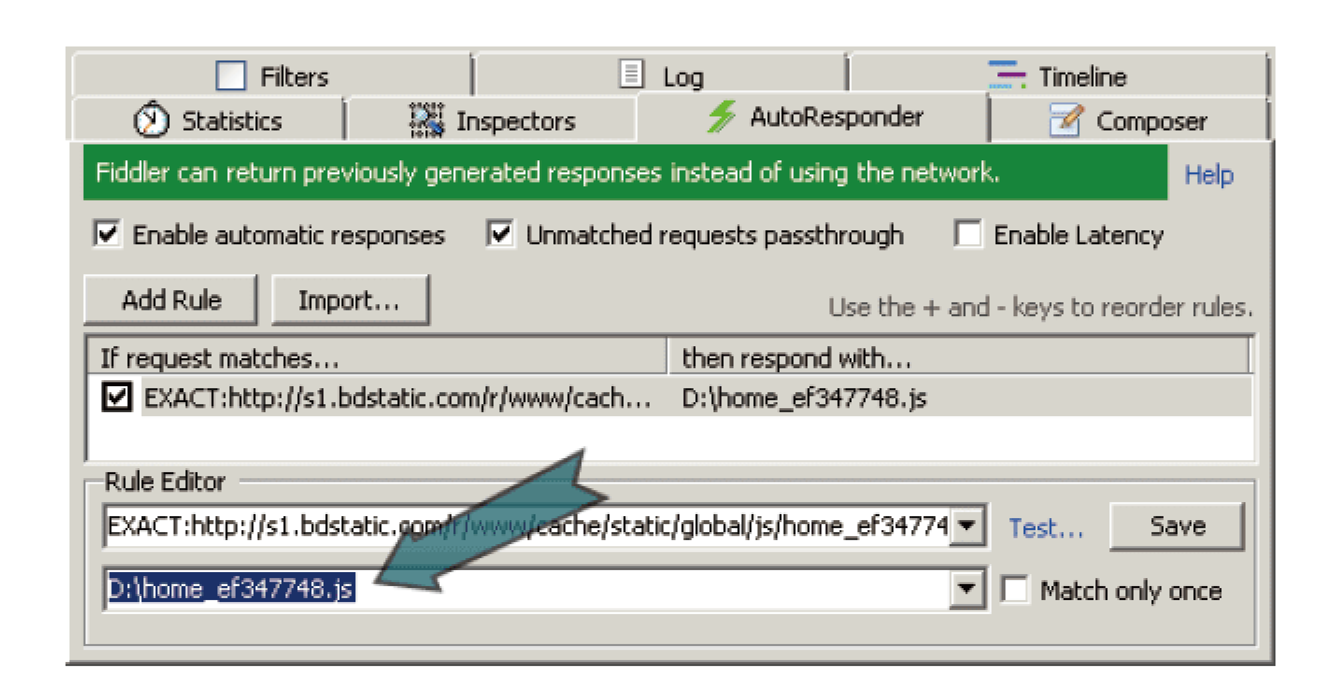


图4.48 修改请求的本地代理地址

（7）打开Chrome浏览器，在地址栏内输入http://www.baidu.com并按Enter键进入，此时百度首页变为黑色，Fiddler代理成功。



示例中完成的功能只是一种简单的模拟，读者在实际开发中可以通过Fiddler代理，修改本地脚本的具体代码，再结合Chrome浏览器的调试功能，解决网站的线上问题将变得简单且高效。

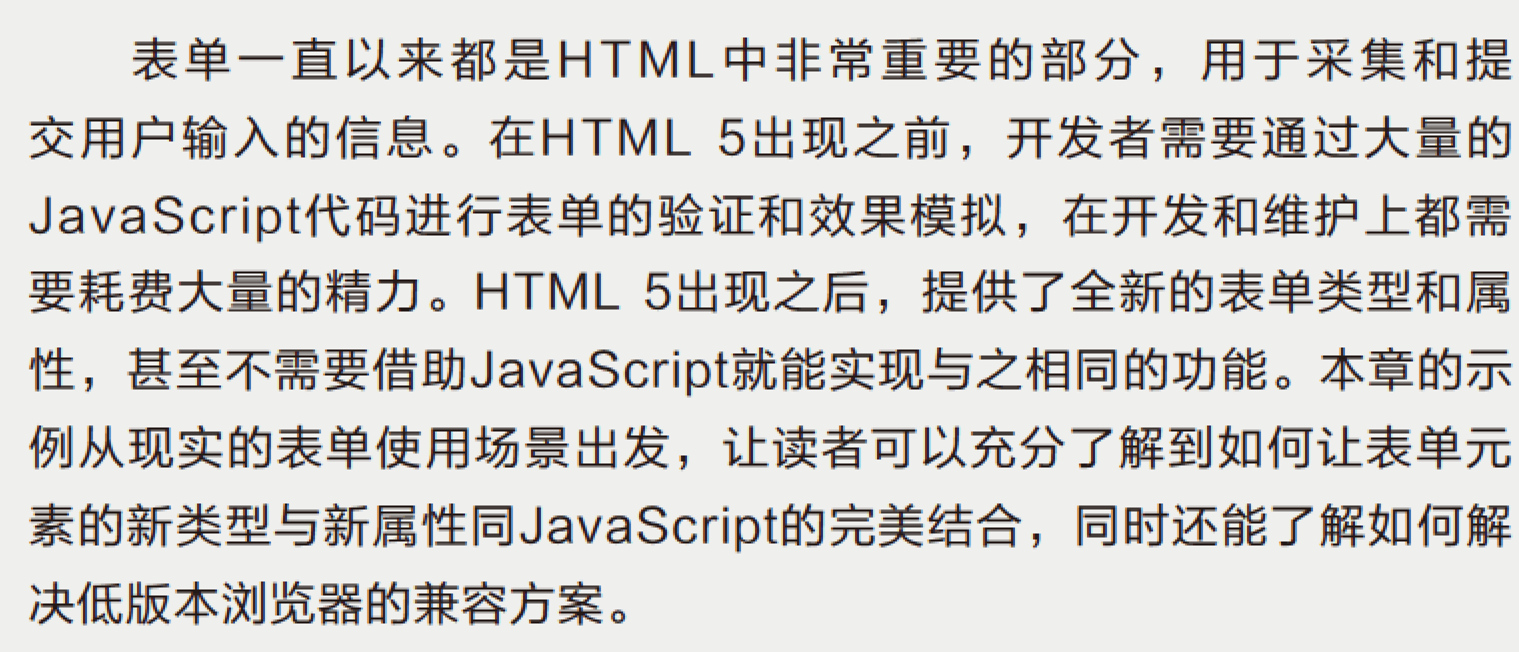


4.5 本章小结

本章介绍了本书示例运行所需要的软件环境，包含编辑器、Node.js、jQuery，同时还介绍了如何用Chrome浏览器进行开发测试，使用Fiddler进行请求代理，提升软件开发效率。学习Node.js有助于了解JavaScript在非浏览器环境下的工作方式，同时还能快速编写Web应用服务，全面了解HTML 5在实际开发中的前后端实现。jQuery框架是目前市面上最火热的JavaScript框架，通过该框架的学习，读者可以轻松地对浏览器DOM结构进行操作，而不用考虑不同浏览器的兼容情况。本章最后的实战技巧，有助于读者提高实际前端开发的工作效率，相信读者在学习完本章之后，通过反复地练习，必定能受益匪浅。



第5章 HTML 5元素与表单大演练



表单一直以来都是HTML中非常重要的部分，用于采集和提交用户输入的信息。在HTML 5出现之前，开发者需要通过大量的JavaScript代码进行表单的验证和效果模拟，在开发和维护上都需要耗费大量的精力。HTML 5出现之后，提供了全新的表单类型和属性，甚至不需要借助JavaScript就能实现与之相同的功能。本章的示例从现实的表单使用场景出发，让读者可以充分了解到如何让表单元素的新类型与新属性同JavaScript的完美结合，同时还能了解如何解决低版本浏览器的兼容方案。

本章知识点：

- 各种新的表单元素
- 跨浏览器的HTML 5表单
- 文件与图片上传
- 多文件上传与异步上传



5.1 示例1 创建跨浏览器的HTML 5表单

5.1.1 示例效果

本示例是一个有关用户详细信息的表单。用户可以在表单内填写姓名、年龄、生日、个人状态、幸运色、博客、邮箱，填完上述的信息后单击“提交”按钮进行保存。示例中运用多种input的新属性和类型，如新属性有placeholder、required、min、max、step，新类别有number、date、range、color、url、email。在不支持新特性的浏览器上使用jQuery UI、ColorPicker、Placeholder 、Web Form 2第三方组件进行模拟。

使用支持HTML 5表单新特性的浏览器Chrome打开，运行效果如图5.1所示。单击“生日”右侧朝下方的小三角，运行效果如图5.2所示。单击“幸运色”右侧颜色选择器，运行效果如图5.3所示。单击“提交”按钮，表单自动进行输入校验，对未满足输入要求的进行错误提示，运行效果如图5.4所示。



图5.1　创建跨浏览器的HTML 5表单效果图



图5.2　单击“生日”右侧朝下方的小三角效果



图5.3　单击“幸运色”右侧红色颜色选择器效果



图5.4　单击“提交”按钮后效果

这里使用不支持新特性的浏览器IE 8来打开网页文件，运行效果如图5.5所示。单击“提交”按钮，运行效果如图5.6所示。



图5.5　表单在IE 8下的运行情况

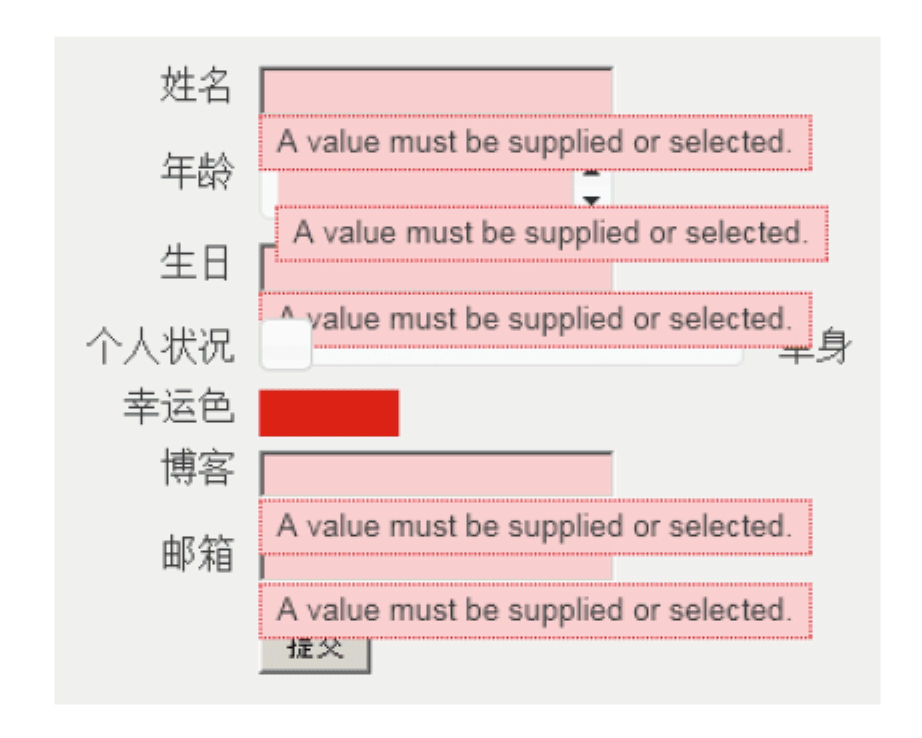


图5.6　在IE 8下单击“提交”按钮

5.1.2 代码设计

编辑如下代码，并保存为“001.创建跨浏览器的HTML 5表单.html”文件，代码如下：

```
<! DOCTYPE HTML>
<html>
<head>
    <link rel="stylesheet" href="../css/jquery-ui-1.9.2.custom.css">
    <link rel="stylesheet" href="../css/colorpicker.css">
    <link rel="stylesheet" href="../css/webforms2.css">
    <link rel="stylesheet" href="../css/html5forms.layout.css">
    <script src="../js/jquery-1.8.3.js"></script>    <!-- 依赖的脚本 --->
    <script src="../js/jquery-ui-1.9.2.custom.js"></script>
    <script src="../js/jquery.placeholder.js"></script>
    <script src="../js/colorpicker.js"></script>
    <script src="../js/modernizr.custom.2.6.2.js"></script>
    <script src="../js/webforms2_src.js"></script>
</head>
<body>
    <header>
        <h2>创建跨浏览器的HTML 5表单</h2>
    </header>
    <form>
    <div>
        <label for="placeholder">姓名</label>
        <input id="placeholder" name="placeholder" type="text"
placeholder="请输入" required />
    </div>
    <div>
        <label for="spinner">年龄</label>
        <span class="age">
            <input id="spinner" name="spinner" type="number" min="1"
max="100" step="1" required />
        </span>
    </div>
```

```
    <div>
        <label for="datepicker">生日</label>
        <input id="datepicker" name="datepicker" type="date" required />
    </div>
    <div>
        <label for="range">个人状况</label>
        <input id="range" name="range" type="range" min="1" max="5" value="1" step="1" />
        <div id="slider" style="display: none;" class="slider" min="1" max="5" value="1" step="1"></div>
        <span id="state" class="tip">单身</span>
    </div>
    <div>
        <label for="colorpicker">幸运色</label>
        <input id="colorpicker" name="colorpicker" type="color" value="#e71605" />
        <span id="colorpicker-span"></span>
    </div>
    <div>
        <label for="url">博客</label>
        <input type="url" id="url" name="url" placeholder="http://www.xxx.com" required />
    </div>
    <div>
        <label for="email">邮箱</label>
        <input type="email" id="email" name="email" placeholder="xxx@xxx.com" required />
    </div>
    <button type="submit">提交</button>
    </form>
</body>
<script>
    //个人状况数据枚举
    var STATE_HASH = {
        '1': '单身',
        '2': '热恋',
        '3': '备婚',
        '4': '已婚',
        '5': '保密'
    };
    if (!Modernizr.inputtypes.number) {     //判断input是否为number类型
        $("#spinner").spinner();
    };
    if (!Modernizr.inputtypes.date) {      //判断input是否为date类型
        $("#datepicker").datepicker();
    };
    if (!Modernizr.input.placeholder) {     //判断input是否为placeholder类型
        $("#placeholder").addClass('placeholder').placeholder();
        $("#url").addClass('placeholder').placeholder();
        $("#email").addClass('placeholder').placeholder();
```

```
75      };
76      if (!Modernizr.inputtypes.color) {          //判断input是否为color类型
77          $("#colorpicker").hide();               //隐藏color文本输入框
78          //初始化ColorPicker组件并添加确认回调函数onSubmit
79          $("#colorpicker-span").ColorPicker({
80              onSubmit: function (hsb, hex, rgb, el) {
81                  $(el).css('background', '#' + hex);
82              }
83          }).show();
84      };
85      if (!Modernizr.inputtypes.range) {          //判断input是否为range类型
86          $('#range').hide();                     //隐藏range文本输入框
87          $('#slider').show().slider({    //使用jQuery UI初始化slider控件
88              min: 1,                             //拖动最小值
89              max: 5,                             //拖动最大值
90              step: 1,                            //拖动间隔
91              change: function (event, ui) {
92                  //获取拖动的值，从个人状况枚举中获取对应类型
93                  $("#state").html(STATE_HASH[ui.value]);
94              }
95          });
96      } else {
97          $('#range').on('change', function () {
98              //从个人状况枚举中获取对应类型
99              $("#state").html(STATE_HASH[$(this).val()]);
100           });
101     };
102   </script>
103 </html>
```

代码利用Modernizr库判断浏览器是否支持HTML 5表单特性，如果支持则采用默认的浏览器行为，如果不支持则调用jQuery插件模拟特性。

5.1.3 代码分析

代码第01行声明了文档所使用的HTML规范，读者可以看到HTML 5对DOCTYPE做了极大的简化，体现了化繁为简的设计准则。对比下HTML 4的DOCTYPE代码：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">
```

示例中的JavaScript代码统一使用Modernizr来检测当前浏览器是否支持文本框类型，利用Modernizr.inputtypes[type]对input类型进行检测。

第58行~第64行是一个数据枚举，列出用户状态的5种类型。接下来的代码由5个if组成，用来对表单元素初始化。

第65行~第67行用于判断浏览器是否支持input的number类型，如果不支持则使用jQuery UI的数字调节控件。

第68行~第70行用于判断浏览器是否支持input的date类型，如果不支持则使用jQuery UI日期控件。

第71行~第75行用于判断浏览器是否支持input的placeholder属性，如果不支持则使用第三方jQuery的插件生成，并给每个元素添加placeholder样式名（用于非焦点情况下的字体变灰）。

第76行~第84行用于判断浏览器是否支持input的color类型。如果不支持则使用Stefan Petre的colorpicker插件实现颜色选择器，并传入onSubmit函数用于监听组件提交，代码如下所示：

```
onSubmit: function (hsb, hex, rgb, el) {
    $(el).css('background', '#' + hex);
}
```

第85行~第101行用于判断浏览器是否支持input的range类型。如果不支持则隐藏自身元素，显示相邻div元素，调用slider插件进行初始化，传入参数。如果支持则监听元素的change事件，在用户拖动滑块时动态地改变个人状态提示。



采用div作为jQuery UI的滑块组件元素，是由于插件不支持input标签调用。

5.1.4 相关知识

1. <input>标签

input除了上面出现的number、date、text、color、range、url、email这7种类型外，还有password、checkbox、file、hidden等老类型，同时新增了time、month、week、datetime等新类型。

2. Web Forms 2.0

示例中引用了webforms2_src.js脚本，该类库提供HTML 5表单之前的版本“WHATWG 网络表单 2.0”规格的跨浏览器实现，用户使用旧的浏览器访问使用了新特性的页面也能享受到相同的体验。详见GitHub项目地址https://github.com/westonruter/webforms2。



5.2 示例2 搞定低版本浏览器的兼容性问题

5.2.1 示例效果

该示例为读者展现了一个典型的优雅降级场景。例子使用HTML 5的Geolocation功能获取用户当前地理位置信息，然后将这个信息作为标记显示在谷歌地图控件上，对于不支持的旧版本浏览器将提供一个表单，用户可以在上面输入地址，调用谷歌提供的查询API获取经纬

度标注在地图上。

在打开示例前请确保机器已经连接上了互联网，使用最新版Firefox打开网页文件。



这里不使用Chrome的原因是由于本地运行示例用Chrome浏览器无法得到获取当前经纬度的权限，需把文件放在本地架设的Web服务器上，如Apache、Nginx、IIS等。

运行效果如图5.7所示。



图5.7 搞定低版本浏览器的兼容性

单击“显示当前位置”按钮，Firefox浏览器会提醒是否“共享方位信息”，确认以后效果如图5.8所示。



图5.8 单击“共享方位信息”显示当前位置

接下来使用不支持Geolocation功能的浏览器IE 8打开示例，运行效果如图5.9所示。



图5.9 IE 8下运行情况

在文本框内输入“上海”，单击“定位当前位置”，运行效果如图5.10所示。



图5.10 IE 8下输入结果查询

5.2.2 代码设计

利用编辑器编辑如下代码，并保存为“002.搞定低版本浏览器的兼容性.html”文件。

```
<! DOCTYPE HTML>
<html>
<head>
    <title>搞定低版本浏览器的兼容性</title>
    <style>
        input
        {
            width:80px;
        }
        /* "显示当前位置" 按钮样式*/
        #see-position
        {
            margin-bottom: 10px;
```

```
        }
        /* 地图容器样式 */
        #geo-map
        {
            height: 280px;
            width: 640px;
        }
        /*提示信息样式*/
        #geo-log
        {
            background-color: #cd4949;
            color: white;
            padding: 5px;
            border-radius: 5px;
            visibility: hidden;
        }
    </style>
    <script src="../js/modernizr.js"></script>
    <script src="http://maps.google.com/maps/api/js?sensor=true">
</script>
    <script src="http://www.google.com/uds/api?file=uds.js&amp;v
=1.0&amp;key=ABQIAAAAjU0EJWnWPMv7oQ-jjS7dYxQ82LsCgTSsdpNEnBsExtoeJv4cd
BSUkiLH6ntmAr_5O4EfjDwOa0oZBQ"
        type="text/javascript"></script>
</head>
<body>
    <header>
        <h2>搞定低版本浏览器的兼容性</h2>
    </header>
    <section>
        <button id="see-position">显示当前位置</button>
        <span id="geo-log"></span>
        <div id="geo-map" class="gmap example">
            <img
src="http://maps.google.com/maps/api/staticmap?center=35.86166,104.195
397&zoom=5&size=680x278&sensor=true" />
        </div>
    </section>
</body>
<script>
    Modernizr.load({
        test: Modernizr.geolocation,    // 判断浏览器是否支持Geolocation
        yep: '../js/lz-geo.js',         // 支持Geolocation加载脚本
        nope: '../js/lz-geo-polyfill.js'// 不支持Geolocation加载脚本
    });
</script>
</html>
```

代码利用Modernizr库判断浏览器是否支持Geolocation特性加载不同脚本，同时还引用了谷歌地图脚本和谷歌地图搜索控件脚本。

利用编辑器编辑如下代码，并保存为lz-geo.js文件。

```
(function (W) {
    var
    map = null,
    geo_log = W.querySelector("#geo-log"),              // 提示信息div
    geo_map = W.querySelector("#geo-map"),              // 地图div
    geo_button = W.querySelector('#see-position'),     // 查询按钮
    geo = {
        //业务初始化
        init: function () {
            geo_button.addEventListener('click', this.successPositionHandler, false);
            geo_map.addEventListener('click', this.successPositionHandler, false);
        },
        successPositionHandler: function () { //成功获取用户位置信息回调
            // 当用户单击时，如果地图不存在，则载入地图（默认中国地图）
            var self = geo;
            if (!map) {
                map = new google.maps.Map(geo_map, {
                    zoom: 3,
                    center: new google.maps.LatLng(35.86166, 104.195397),
                    mapTypeId: google.maps.MapTypeId.ROADMAP
                });
                map.getDiv().style.border = '1px solid #ccc';
            };
            geo_log.style.visibility = 'visible';
            geo_log.textContent = '查找当前位置';
            navigator.geolocation.getCurrentPosition(self.showPosition, self.handlePositionError);
        },
        showPosition: function (position) {            //显示当前位置
            geo_log.textContent = "你现在的位置在经纬度（" + position.coords.latitude + ", " +
                position.coords.longitude + "）" + position.coords.accuracy + "米的范围内";
            //在Google地图上显示位置
            var latLng = new google.maps.LatLng(position.coords.latitude, position.coords.longitude);
            var marker = new google.maps.Marker({ position: latLng, map: map });
            map.setCenter(latLng);
            map.setZoom(15);
        },
        handlePositionError: function (error) { //当前位置信息获取错误，打印出错误信息
            geo_log.textContent = error.message;
        }
    }
```

```
    ;
    geo.init();
})(document);
```

利用编辑器编辑如下代码，并保存为lz-geo-polyfill.js文件。

```
(function (W) {
    var
    map = null,
      search_key,
      gLocalSearch,
      geo_log = W.getElementById("geo-log"),         // 提示信息div
      geo_map = W.getElementById("geo-map"),         // 地图div
      geo_button = W.getElementById('see-position')  // 查询按钮
    ;
    function addEvent(evnt, elem, func) {
        if (elem.addEventListener)
            elem.addEventListener(evnt, func, false);
        else if (elem.attachEvent)
            elem.attachEvent("on" + evnt, func);
        else
            elem[evnt] = func;
    };
    function insertAfter(newEl, targetEl) {
        var parentEl = targetEl.parentNode;
        if (parentEl.lastChild == targetEl)
            parentEl.appendChild(newEl);
        else
            parentEl.insertBefore(newEl, targetEl.nextSibling);
    };
    var geo = {
        //业务初始化
        init: function () {
            addEvent('click', geo_button, this.successPositionHandler);
            addEvent('click', geo_map, this.successPositionHandler);

            gLocalSearch = new GlocalSearch();
            gLocalSearch.setSearchCompleteCallback(null,
this.showPosition);
            this.ui();
        },
        ui: function () {
            geo_button.innerText = '定位当前位置';
            search_key = document.createElement('input');
            insertAfter(search_key, geo_button);
        },
        successPositionHandler: function () {
            // 当用户单击时，如果地图不存在，则载入地图（默认中国地图）
            var self = geo;
            if (!map) {
                map = new google.maps.Map(geo_map, {
                    zoom: 3,
                    center: new google.maps.LatLng(35.86166,
104.195397),        // 中国
                    mapTypeId: google.maps.MapTypeId.ROADMAP
```

```
48                  });
49                  map.getDiv().style.border = '1px solid #ccc';
50              };
51              geo_log.style.visibility = 'visible';
52              geo_log.innerText = '查找当前位置';
53              gLocalSearch.execute(search_key.value);
54          },
55          //显示当前位置
56          showPosition: function () {
57              var first = gLocalSearch.results[0];
58              geo_log.innerText = "你现在的位置在经纬度（" + first.lat + ", "
59  +first.lng + "）" + first.accuracy + "米的范围内";
           //在Google地图上显示位置
60              var latLng = new google.maps.LatLng(first.lat, first.lng);
61              var marker = new google.maps.Marker({ position: latLng,
    map: map });
62              map.setCenter(latLng);
63              map.setZoom(15);
64          }
65      };
66      geo.init();
67  })(document);
```

5.2.3 代码分析

首先看HTML文件中的脚本代码，如下所示：

```
 Modernizr.load({
        test: Modernizr.geolocation,
        yep: '../js/lz-geo.js',
        nope: '../js/lz-geo-polyfill.js'
 });
```

Modernizr.load方法用作对资源进行动态加载。示例中实现了两个Polyfill，判断浏览器是否支持Geolocation特性，如果是则加载lz-geo.js脚本，不是则加载lz-geo-polyfill.js脚本。



Polyfill是旧浏览器上HTML 5技术或功能API的JavaScript补充，用于动态的加载代码或库，在不支持的浏览器中模拟HTML 5新特性。GitGub上有个项目收集了目前市面上拥有的Polyfills，链接地址为https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills。

接着分析脚本lz-geo.js。初始化函数init，分别在地图容器和“显示当前位置”按钮上绑定successPositionHandler事件。见lz-geo.js文件第9行~第12行。

在successPositionHandler方法中，用户单击“显示当前位置”按钮或者地图容器时，如果地图实例不存在，则调用谷歌地图提供的方法进行地图对象实例化，并设置经纬度（中心区域为中国），再调用navigator.geolocation.getCurrentPosition方法，传入两个回调函数，函数分别在成功和失败时响应接收。见lz-geo.js文件第13行~第27行。

在showPosition方法中，该方法作为回调函数传入navigator.geolocation.getCurrentPosition，当浏览器获取用户经纬度成功后产生回调，回调函数接收一个position对象，从该对象中获取精准度和经纬度进行展现，并在谷歌地图上标注。见lz-geo.js文件第28行~第36行。

在handlePositionError方法中，浏览器获取用户当前经纬度失败时回调该方法，回调函数接收一个error对象，获取错误信息并在页面上进行提示，代码如下：

```
handlePositionError: function (error) {
    geo_log.textContent = error.message;
}
```

分析完lz-geo.js脚本后，看看当浏览器不支持Geolocation时是如何处理的，代码在lz-geo-polyfill.js文件中。

初始化函数init的前两段代码和之前相同，增加了一个对gLocalSearch的初始化，同时在其实例上绑定查询结束的监听事件，并调用ui方法为用户插入一个输入框，对不兼容新特性的浏览器进行优雅降级处理。见lz-geo-polyfill.js文件第27行~第39行。

successPositionHandler方法与之前不同，不支持新特性的浏览器会加载谷歌地图搜索控件脚本。函数调用gLocalSearch实例的execute方法，传入用户输入的地址，运用JSONP从谷歌服务器端获取可能的经纬度列表，代码如下：

```
gLocalSearch.execute(search_key.value);
```

showPosition方法已经在初始化函数中绑定在gLocalSearch实例的查询回调事件上，execute执行查询后该方法被回调，同时在gLocalSearch实例的result属性上得到搜索结果，取最靠前的返回结果，并展现在谷歌地图上。见lz-geo-polyfill.js文件的第56行~第64行。

5.2.4 相关知识

1. getCurrentPosition方法

该方法用于定位用户的位置。目前Chrome、IE 9、Firefox、Safari、Opera支持地理定位。对于拥有GPS的设备，比如iPhone，地理定位将更加精确。详细的使用可以参考http://www.w3school.com.cn/html5/html_5_geolocation.asp中的内容。

2. JSONP

JSONP是一个跨域进行数据访问传输的协议。原理就是通过动态创建script元素指向数据服务器，利用JavaScript callback的形式实现跨域访问。



示例3 创建HTML 5版的注册页面

5.3.1 示例效果

本示例是一个常见的用户注册页面，表单由三个文本框组成，类型分别为email、text和

password。进入页面后，“电子邮箱”文本框将自动获得焦点，同时文本框的边框会产生红色渐变的效果。单击“昵称”文本框，“电子邮件”文本框的边框红色消失，此时“昵称”文本框的边框出现红色渐变效果，“密码”文本框的效果相同。

使用支持HTML 5表单新特性的浏览器Chrome打开网页文件，运行效果如图5.11所示。

打开网页的同时，“电子邮箱”文本框会渐变为红色，运行效果如图5.12所示。



图5.11　HTML 5版的注册页面



图5.12　“电子邮箱”文本框获得焦点

单击“注册”按钮，进行表单验证，运行效果如图5.13所示。



图5.13　单击“注册”按钮

示例中采用了大量的CSS 3特效。刚进入页面聚焦渐变的动画、表单背景色的颜色渐变、文本框的阴影等。通过本示例可以了解如何运用CSS 3制作简单的注册页面。

5.3.2 代码设计

利用编辑器编辑如下代码，并保存为“003.创建HTML 5版的注册页面.html”文件。

```
<!doctype html>
<html>
<head>
    <style>
        *:focus{ outline: none; }       /* 所有元素焦点样式 */
        body { text-align: center; }
        form                            /* 表单样式 */
        {
```

```
            height: 240px;
            width: 400px;
            margin: -200px 0 0 -240px;
            padding: 30px;
            position: absolute;
            top: 50%;
            left: 50%;
            z-index: 0;
            background-color: #eee;

            /* gradient带webkit前缀被 Safari 4+, Chrome 支持 */
            background-image: -webkit-gradient(linear, left top,
left bottom, from(#fff), to(#eee));
            /* linear-gradient带webkit前缀被 Chrome 10+, Safari 5.1+,
iOS 5+ 支持 */
            background-image: -webkit-linear-gradient(top, #fff, #eee);
            /* linear-gradient带moz前缀被 Firefox 3.6-15 支持 */
            background-image: -moz-linear-gradient(top, #fff, #eee);
            /* linear-gradient带ms前缀被 IE9+ 支持 */
            background-image: -ms-linear-gradient(top, #fff, #eee);
            /* linear-gradient带o前缀被 Opera 10.5-12.00 支持 */
            background-image: -o-linear-gradient(top, #fff, #eee);
            /* 标准格式linear-gradient被 Opera 10.5, IE9+, Safari 5,
 Chrome, Firefox 4+, iOS 4, Android 2.1+ 支持 */
            background-image: linear-gradient(top, #fff, #eee);
            /* border-radius带moz前缀被 Firefox 3.5+ 支持 */
            -moz-border-radius: 3px;
            /* border-radius带webkit前缀被 Safari 3-4, iOS 1-3.2,
Android ≤1.6 支持*/
            -webkit-border-radius: 3px;
            /* 标准格式border-radius被 Opera 10.5, IE9+, Safari 5,
 Chrome, Firefox 4+, iOS 4, Android 2.1+支持 */
            border-radius: 3px;

            /* box-shadow带webkit前缀被 Safari 3-4, iOS 4.0.2 - 4.2,
Android 2.3+ 支持 */
            -webkit-box-shadow:  0 0 2px rgba(0, 0, 0, 0.2), 0 1px 1px
 rgba(0, 0, 0, .2), 0 3px 0 #fff, 0 4px 0 rgba(0, 0, 0, .2) ;
            /* box-shadow带moz前缀被 Firefox 4+ 支持 */
            -moz-box-shadow:  0 0 2px rgba(0, 0, 0, 0.2),
0 1px 1px rgba(0, 0, 0, .2), 0 3px 0 #fff, 0 4px 0
 rgba(0, 0, 0, .2) ;
            /* 标准格式box-shadow被 Opera 10.5, IE9+, Firefox 4+,
Chrome 6+, iOS 5 支持*/
            box-shadow: 0 0 2px rgba(0, 0, 0, 0.2), 0 1px 1px rgba(0,
 0, 0, .2), 0 3px 0 #fff, 0 4px 0 rgba(0, 0, 0, .2) ;
        }
        form:before /* before伪元素表示在一个元素的内容之前插入content属性
定义的内容与样式*/
        {
            content: ''; /* content属性与:befor及:after伪元素配合使用，生成
```

```
某个 CSS 选择器之前或之后的内容 */
            position: absolute;
            z-index: -1;
            border: 1px dashed #ccc;
            top: 5px;
            bottom: 5px;
            left: 5px;
            right: 5px;

            -moz-box-shadow: 0 0 0 1px #fff;    /* Firefox 4+ */
            -webkit-box-shadow: 0 0 0 1px #fff;  /* Safari 3-4,
iOS 4.0.2 - 4.2, Android 2.3+ */
            box-shadow: 0 0 0 1px #fff;         /* Opera 10.5, IE9+,
Firefox 4+, Chrome 6+, iOS 5 */
        }
        input    /* 所有文本框样式 */
        {
            float: left;
            padding: 15px 15px 15px 45px;
            margin: 0 0 10px 0;
            width: 353px;
            border: 1px solid #CCC;
            background: #F1F1F1;
            font-size: 14px;
            -webkit-border-radius: 5px;
            -moz-border-radius: 5px;
            border-radius: 5px;

            -moz-border-radius: 5px;    /* Firefox 3.5+ */
            -webkit-border-radius: 5px;  /* Safari 3-4, iOS 1-3.2,
 Android ≤1.6 */
            border-radius: 5px;         /* Opera 10.5, IE9+, Safari 5,
 Chrome, Firefox 4+, iOS 4, Android 2.1+ */
            -moz-box-shadow: 0 1px 1px #ccc inset, 0 1px 0 #fff;
/* inset表示盒内阴影; Firefox 4+ */
            -webkit-box-shadow: 0 1px 1px #CCC inset, 0 1px 0 white;/*
Safari 3-4, iOS 4.0.2 - 4.2, Android 2.3+ */
            box-shadow: 0 1px 1px #CCC inset, 0 1px 0 white;
 /* Opera 10.5, IE9+, Firefox 4+,
Chrome 6+, iOS 5 */

            /* ease(逐渐慢下来);linear(匀速);ease-in(由慢到快);ease-out(由
快到慢);ease-in-out(先慢到快再到慢) */
            -webkit-transition: all 0.5s ease-in-out;
/* Safari 3.2+, Chrome */
            -moz-transition: all 0.5s ease-in-out;  /* Firefox 4-15 */
            -o-transition: all 0.5s ease-in-out; /* Opera 10.5-12.00 */
            transition: all 0.5s ease-in-out;   /* Firefox 16+,
Opera 12.50+ */
        }
        input:focus         /* 所有文本框焦点样式 */
```

```
{
    background-color: #fff;
    border-color: #e8c291;
    outline: none;
    -moz-box-shadow: 0 0 0 1px #e8c291 inset;
    -webkit-box-shadow: 0 0 0 1px #e8c291 inset;
    box-shadow: 0 0 0 1px #e8c291 inset;
}
input:hover          /* 所有文本框鼠标悬停样式 */
{
    border-color: inherit !important;
    background-color: #EfEfEf;
    -webkit-border-radius: 5px 0 0 5px;
    -moz-border-radius: 5px 0 0 5px;
    border-radius: 5px 0 0 5px;
}
input:not(:focus) { opacity: 0.6; }   /* 所有文本框非焦点样式 */
input:valid { opacity: 0.8; }         /* 所有文本框输入有效样式 */
input:focus:invalid          /* 所有文本框获得焦点但输入无效样式 */
{
    border: 1px solid red;
    background-color: #FFEFF0;
}
section  { width: 400px; margin: 0 auto; }  /* 章节样式 */
.clearfix { clear: both; }                 /* 清除浮动样式 */
#submit:hover,                  /* 提交按钮鼠标悬停和焦点样式 */
#submit:focus
{
    background-color: #FDDB6F;

    /* 可以参考form样式中的gradient注释 */
    background-image: -webkit-gradient(linear, left top, left bottom, from(#FFB94B), to(#FDDB6F));
    background-image: -webkit-linear-gradient(top, #FFB94B, #FDDB6F);
    background-image: -moz-linear-gradient(top, #FFB94B, #FDDB6F);
    background-image: -ms-linear-gradient(top, #FFB94B, #FDDB6F);
    background-image: -o-linear-gradient(top, #FFB94B, #FDDB6F);
    background-image: linear-gradient(top, #FFB94B, #FDDB6F);
}
#submit             /* 提交按钮样式 */
{
    background-color: #FFB94B;
    border-width: 1px;
    border-style: solid;
    border-color: #D69E31 #E3A037 #D5982D #E3A037;
    float: left;
    height: 35px;
```

```
            padding: 0;
            width: 120px;
            cursor: pointer;
            font: bold 15px Arial, Helvetica;
            color: #8F5A0A;
            margin: 20px 0 0 0;

            /* 可以参考form样式中的gradient注释 */
            background-image: -webkit-gradient(linear, left top, 
left bottom, from(#FDDB6F), to(#FFB94B));
            background-image: -webkit-linear-gradient(top, #FDDB6F, 
#FFB94B);
            background-image: -moz-linear-gradient(top, #FDDB6F, 
#FFB94B);
            background-image: -ms-linear-gradient(top, #FDDB6F, 
#FFB94B);
            background-image: -o-linear-gradient(top, #FDDB6F, 
#FFB94B);
            background-image: linear-gradient(top, #FDDB6F, #FFB94B);

            /* 可以参考form样式中的border-radius注释 */
            -moz-border-radius: 3px;
            -webkit-border-radius: 3px;
            border-radius: 3px;

            /* 给文字加上阴影，早在CSS 2中已经出现 */
            text-shadow: 0 1px 0 rgba(255, 255, 255, 0.5);

            /* 可以参考form样式中的box-shadow注释 */
            -moz-box-shadow: 0 0 1px rgba(0, 0, 0, 0.3), 0 1px 
0 rgba(255, 255, 255, 0.3) inset;
            -webkit-box-shadow: 0 0 1px rgba(0, 0, 0, 0.3), 0 1px 
0 rgba(255, 255, 255, 0.3) inset;
            box-shadow: 0 0 1px rgba(0, 0, 0, 0.3), 0 1px 0 rgba
(255, 255, 255, 0.3) inset;
        }
        .item-name { background: url(../images/user.png) 10px 11px 
no-repeat; }    /* 昵称背景样式 */
        .item-email { background: url(../images/email.png) 10px 
11px no-repeat; }   /* 密码背景样式 */
        .item-password { background: url(../images/keys.png) 10px 
11px no-repeat; } /* 电子邮箱背景样式 */
    </style>
    <script src="../js/jquery-1.8.3.js"></script>
    <script src="../js/modernizr.custom.2.6.2.js"></script>
</head>
<body>
    <header><h2>搞定输入框自动聚焦</h2></header>
    <section>
        <form action="" method="post">
            <div class="clearfix">
```

```
182                 <!-- 第1个autofocus -->
183                 <input type="email" tabindex="1"
   id="email" class="item-email" placeholder="电子邮箱" autofocus required/>
184             </div>
185             <div class="clearfix">
186                 <!-- 第2个autofocus -->
187                 <input type="text" tabindex="2" id="name" class="item-
188 name" placeholder="昵称" autofocus required/>
189             </div>
190             <div class="clearfix">
191                 <!-- 第3个autofocus -->
192                 <input type="password" tabindex="3" id="password"
193 class="item-password" placeholder="密码" autofocus autocomplete="off"
194 required/>
195             </div>
196             <div class="clearfix"><input type="submit" tabindex="4"
   id="submit" value="注     册" /></div>
197         </form>
198     </section>
199 </body>
200 </html>
```



在代码中可以看到很多样式前带有-webkit、-moz、-o、-ms的前缀，注释中给出了浏览器的支持情况。想更多地了解各浏览器前缀，可以参考网站http://css3please.com中的内容。

5.3.3 代码分析

本示例侧重从CSS 3出发，介绍如何构建一个注册页面，下面讲解其中用到的CSS 3技巧。

代码19~31行、126~132行、149~155行使用了CSS 3 Gradient。渐变分为线性渐变（Linear Gradients）和径向渐变（Radial Gradients）。这里使用了线性渐变，WebKit内核的浏览器语法如下：

```
-webkit-gradient(<type>, [<point> || <angle>,]? <stop>, <stop> [, <stop>]* )
```

WebKit下Gradient的使用语法如图5.14所示。

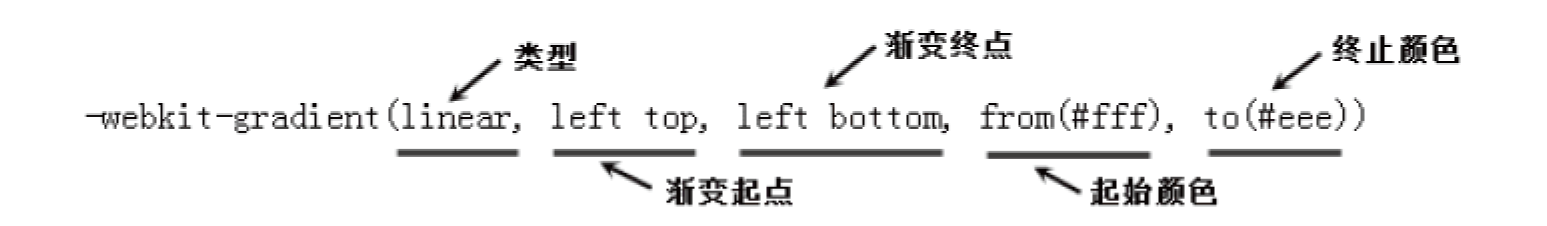


图5.14 WebKit下Gradient使用

第一个参数表示渐变类型，分为linear（线性渐变）和radial（径向渐变）两种。第二个和第三个参数分别表示渐变的起点和终点，可以用坐标形式或方位值，比如right top（右上角）和right bottom（右下角），也可以使用角度，比如red 10%。第四个和第五个参数表示起始和终止的渐变颜色。

标准浏览器的Gradient语法如下：

```
linear-gradient([point || angle,]? stop, stop [, stop]*)
```

标准浏览器下Gradient的使用如图5.15所示。

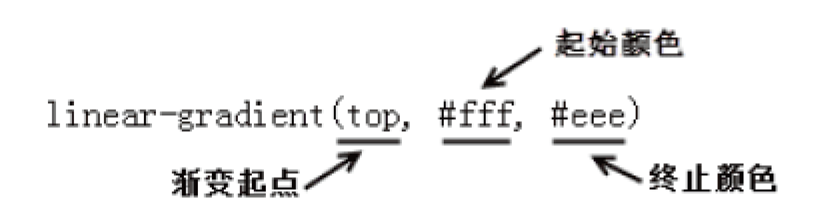


图5.15　标准浏览器下Gradient的使用

标准浏览器下Gradient的渐变类型不在第一个参数上，而是写在样式名称上。第一个参数表示渐变的起点，可以使用方位值或者角度值，第二个和第三个参数和WebKit相同。



想更多了解Gradient的使用，可以参考网站http://css-tricks.com/examples/CSS 3Gradient/。

代码32~38行、79~81行、108~110行、158~160行使用了CSS 3的border-radius，中文意思"圆角"，语法如下：

```
border-radius : none | <length>{1,4} [ / <length>{1,4} ]?
```

其中length是由浮点数字和单位标识符组成的长度值，可以使用em、ex、pt、px、百分比等等，不可为负值。圆角还有其他一些相关属性，如border-top-right-radius，border-bottom-right-radius，border-bottom-left-radius，border-top-left-radius。

代码40~47行、62~64行、82~85行、100~102行、165~168行使用了CSS 3的box-shadow，语法如下：

```
box-shadow : <length> <length> <length> <length> || <color>
```

参数说明：阴影水平偏移值（可取正负值）；阴影垂直偏移值（可取正负值）；阴影边框；阴影模糊值；阴影颜色。

例子中使用的盒阴影效果属于盒外阴影，除此之外还有盒内阴影，使用时增加一个inset，代码如下所示：

```
box-shadow : inset 10px 10px 5px #000000;
```

如果还想了解更多的盒阴影，可以去http://www.css3maker.com/box-shadow.html感受一下box-shadow的强大效果。

5.3.4 相关知识

CSS 3浏览器前缀

浏览器厂商会预先支持一些还处在草案状态下的CSS属性，为了做区分所以在样式名前加上各自的前缀。目前常用的前缀有-webkit、-moz、-ms、-o。假如在项目中想使用最新的CSS 3属性，又不想写那么多前缀，可以使用一个JavaScript的解决方案，该项目地址为https://github.com/LeaVerou/prefixfree/。



5.4 示例4 用HTML 5的验证方法验证注册页面

5.4.1 示例效果

本示例采用HTML 5的验证方法验证通用的注册表单。用户输入注册表单的注册信息，单击“注册”按钮，HTML 5表单调用浏览器内置表单验证方法，当验证不通过时，表单不能提交，界面上提示出错信息。HTML 5通过内置的表单验证，可以完成一般的验证功能。

一般表单验证的主要目的是纠正用户的输入错误信息，并及时提醒用户输入正确的数据。以往的表单验证，需要开发者写一堆JavaScript的逻辑判断。进一步思考，表单验证可以理解为一个系统，HTML 5已经内置了这一套验证机制，其通过8种验证验证机制，验证HTML 5表单。

本示例表单验证包括用户名、电子邮箱、手机号码、生日、QQ号码、个人站点、密码、重复登录密码。



该示例请用Chrome浏览器打开。

使用Chrome浏览器打开示例文件，运行效果如图5.16所示。

验证不通过时，提示信息如图5.17所示。



图5.16 表单验证注册页面

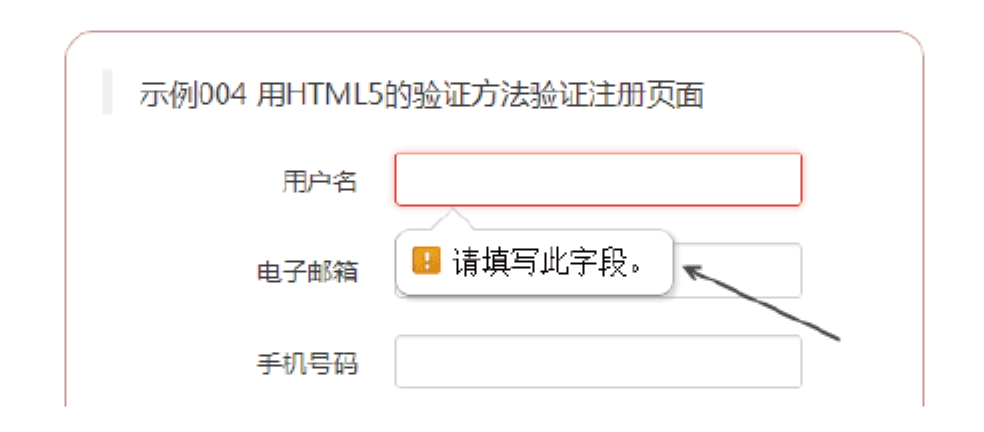


图5.17 用户名为空时的出错提示信息

5.4.2 代码设计

利用编辑器编辑如下代码，并保存为“004.用HTML 5的验证方法验证注册页面.html”文件。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>示例004 用HTML 5的验证方法验证注册页面</title>
    <link rel="stylesheet" href="../css/bootstrap/css/bootstrap.min.css" type="text/css">
    <style type="text/css">
    body{
      margin:50px auto;width:430px;padding:20px;border:1px solid #c88e8e;
      border-radius: 15px;                          <!-- 设置圆角-->
    }
    </style>
</head>
<body>
      <!-- 引用bootstrap的块样式 -->
    <blockquote>
        <p>示例004 用HTML 5的验证方法验证注册页面</p>
    </blockquote><!-- 块结束 -->

    <!-- 表单开始 -->
    <form class="form-horizontal" name="myform">
      <div class="control-group">
            <label class="control-label" for="username">用户名</label>
            <div class="controls">
            <!-- require 在表单控件中将required特性设置为true -->
                <input type="text" name="username" id="username" required>
            </div>
        </div>
        <div class="control-group">
            <label class="control-label" for="email">电子邮箱</label>
            <div class="controls">
            <!-- 指定表单控件的type特性值为email, 设置required为true -->
                <input type="email" name="email" id="email" required>
            </div>
        </div>
        <div class="control-group">
            <label class="control-label" for="phone">手机号码</label>
            <div class="controls">
            <!-- 在表单控件上设置pattern特性，电话号码以13、15或者18开头共11位 -->
                <input type="text" id="phone" pattern="^((13)|18|15)+\d{9}$" name="phone">
            </div>
        </div>
```

```
        <div class="control-group">
            <label class="control-label" for="birth">生日</label>
            <div class="controls">
                 <!-- 指定表单控件的type特性值为date -->
                <input type="date" id="birth" name="birth">
            </div>
        </div>
        <div class="control-group">
            <label class="control-label" for="qq">QQ</label>
            <div class="controls">
            <!-- 在表单控件上设置pattern特性，，设置required为true -->
                <input type="text" required pattern="[1-9][0-9]{4}"
id="qq" name="qq">
            </div>
        </div>
        <div class="control-group">
            <label class="control-label" for="domain">个人站点</label>
            <div class="controls">
            <!-- 指定表单控件的type特性值为url，设置required为true-->
                <input type="url" required id="domain" name="url">
            </div>
        </div>
        <div class="control-group">
            <label class="control-label" for="domain">密码</label>
            <div class="controls">
            <!-- 设置required为true 密码最小长度为6至10位 -->
                <input type="password" required min="6" max="10"
 id="password" name="password">
            </div>
        </div>
        <div class="control-group">
            <label class="control-label" for="domain">重复登录密码
</label>
            <div class="controls">
            <!-- 设置required为true 重复密码长度为6-10位，设置自定义验证规则-->
                <input type="password" required id="repassword" min=
"6" max="10"
oninput="customvalid(this)" name="repassword">
            </div>
        </div>
        <div class="control-group">
            <div class="controls">
                <button type="submit" class="btn">注册</button>
            </div>
        </div>
    </form>
<!-- 表单结束 -->
<script type="text/javascript">
function customvalid(o){
      var v = o.validity,
             //获取重复登录密码的ValidatyStat对象
```

```
                password = document.querySelector("#password");
                //获取登录密码的表单控件
        if(o.value !== password.value){
        //比较密码与重复登录密码的值是否相等
                o.setCustomValidity("两次输入的密码不一致！");
                //两次输入的密码不一致，自定义错误信息
        }else{
                o.setCustomValidity("");
                //验证通过，清除出错信息
        }
}
</script>
</body>
</html>
```

5.4.3 代码分析

1. validity属性

HTML 5引入了8种用于验证表单控件数据正确性的方法，分别是valueMissing（值丢失）、typeMisMatch（类型不匹配）、pattenMismatch（模式不匹配）、tooLong（字符太长）、rangeUnderflow（范围下溢）、rangeOverflow（范围上溢）、stepMismath（步阶不匹配）和customError（自定义错误）。

HTML 5的表单控件新增了一个属性为validity，如下：

```
document.myform.username.validity
```

validity为ValidityState对象，除包含了上述的8个属性外，还有一个属性是valid，当表单验证执行完毕，会返回一个布尔值，它表示表单控件是否已通过了所有的验证约束条件。可以把valid属性看做是最终验证结果，当表单验证的8个约束条件都验证通过，则valid就为true，反之，只要有一项未验证通过，valid就为false。

valueMissing表示确保表单控件的值已填写，使用方法：在表单控件上添加required，比较表单控件的valueMissing是否为true，如果为true，则修改出错信息。如修改“用户名”出错时的出错信息为“用户名不能为空”，代码如下：

```
<!--  HTML代码 -->
用户名：<input type="text" required id="username" name="username"
   onblur="valit_username();"/>
<!-- JavaScript 代码 -->
<script>
    function valit_username(){
          var username = document.querySelector("#username");
          //获取用户名表单控件
          if(true == username.validity.valueMissing){
          //判断用户名是否为空
                 username.setCustomValidity('用户名不能为空');
                 //验证失败，修改提示信息
          }else{
```

```
                username.setCustomValidity('');           //清除出错信息
            }
        }
        valit_username();                                 //出错信息初始化
    </script>
```

typeMismatch表示保证表单控件值与预期类型相匹配，预期类型包括url、email、number。

使用方法：在表单控件上设置type值为url、email、number之一。如验证电子邮件，当电子邮件的格式不正确时，改变“电子邮箱”默认的提示信息为“电子邮箱格式不正确”，代码如下：

```
    <!-- HTML代码 -->
    电子邮箱：<input type="email" required id="email" name="email"
     onblur="valit_email();"/>
    <!-- JavaScript 代码 -->
    <script>
        function valit_email(){
            var email = document.querySelector("#email");
            //获取电子邮箱表单控件
            if(true == email.validity.typeMismatch){   //判断电子邮箱格式是否正确
                email.setCustomValidity('电子邮箱格式不正确');
                //验证失败，修改提示信息
            }else{
                email.setCustomValidity('');              //清除出错信息
            }
        }
        valit_email();
    </script>
```

patternMismatch表示验证表单控件值与其pattern设置的正则表达式相匹配。

使用方法：在表单控件上设置pattern的值为要验证的正则表达式，如验证手机号码是否正确，代码如下：

```
    <!-- HTML代码 -->
    <input type="text" id="phone" required pattern="^((13)|18|15)+\d{9}
    $" name="phone" onblur="valit_phone()">
    <!-- JavaScript 代码 -->
    <script>
        function valit_phone(){
            var phone = document.querySelector("#phone");
            //获取手机号码表单控件
            if(true == phone.validity.typeMismatch){
            //判断手机号码是与正则表达式匹配
            //验证失败，修改提示信息
                phone.setCustomValidity('手机号码必须是以13、15、18开头的11位
                数字');
            }else{
                phone.setCustomValidity('');
                      //清除出错信息
```

```
            }
        }
        valit_phone();
    </script>
```

tooLong表示限制最长的字符数，在表单控件上设置maxlength属性值，使用代码如下：

```
<input type="text" maxlength="30" />
```



虽然表单控件通常会在用户输入时限制最大长度（如maxLength设为20，用户无法输入第21个字符），但在有些情况下（如通过程序设置），还是会超出最大值。

rangeUnderflow表示限制数字控件的最小值，在表单控件上设置min属性的值，使用代码如下：

```
<input type="range" min="6" />
```



一般该属性比较少用，但在需要做数值范围检查的表单控件中，数值很可能会暂时低于设置的下限。如果通过程序设置表单控件的值小于min的值，浏览器会自动把该值设置为最小值6。

rangeOverflow表示限制数字控件的最大值，在表单控件上设置max属性的值，使用代码如下：

```
<input type="range" max="50" />
```



如果通过程序设置表单控件的值为max的值，浏览器会自动把该值设置为最大值50。

stepMismath表示确保输入值符合min、max及step的步阶，在表单控件上设置step属性的值，使用代码如下：

```
<input type="range" step="10" min="5" max="200" />
```



如果通过程序设置表单控件的值小于min值，浏览器会自动把该值设置为最小值5。如果大于195，则浏览器会把该值设置为195。

customError表示自定义验证方法。

当以上7种验证模式的出错信息无法满足业务需求，可以使用自定义的出错信息。在表单控件上设置自定义验证出错信息，代码如下：

```
document.querySelector("#password"). setCustomValidity("两次输入的密码不一致！");
```



当验证通过后，需要手动清除出错信息。

```
document.querySelector("#password"). setCustomValidity("");
```

2．示例代码分析

第6行代码引入Bootstrap CSS库美化界面，代码如下：

```
<link rel="stylesheet" href="../css/bootstrap/css/bootstrap.min.css"
 type="text/css">
```



更多Bootstrap知识，请参考网站https://twitter.github.com/bootstrap。

第26行设置“用户名”为必填项，“用户名”为空时，验证失败，其valueMissing值返回true，代码如下：

```
<input type="text" name="username" id="username" required>
```



读者可通过Chrome浏览器控制台输入document.myform.username.validity.valueMissing查看valueMissing值，验证通过时，其值为false，验证不通过其值为true。

第33行设置“电子邮箱”的类型为email，“电子邮箱”为必填项。其实是先验证必填，再验证是否是有效的电子邮箱。当该项值为空时，valueMissing返回true；当不符合电子邮箱格式时，typeMismatch返回true。



读者可通过Chrome浏览器控制台输入document.myform.email.validity.typeMismatch查看typeMismatch值。

第40行设置“手机号码”验证方式为正则表达式验证，“手机号码”必须是以13、15、18开头的11位数字，当手机号码不合法时，patternMismatch返回true，代码如下：

```
<input type="text" id="phone" pattern="^((13)|18|15)+\d{9}$" name="phone">
```



读者可通过Chrome浏览器控制台输入 document.myform.email.validity.patternMismatch查看patternMismatch值。

第47行设置“生日”为日期型，如果输入为非日期型数据，当鼠标离开输入框后，输入框清空。

第61行设置“个人站点”的类型为url，当输入数据为不合法的url时，验证不通过，typeMismatch 返回true。

第68行设置“密码”的验证规则为必填项，最小位数为6位，最大位数为10位，当有一项验证不通过，则提示出错。

第75行和第76行除了与“密码”一样的验证规则外，同时定义了oninput事件用来自定义验证规则，当“重复登录密码”与“密码”不一致时，提示出错信息。

第85行~第93行，定义customvalid函数，当“重复登录密码”与“密码”不一致时，用setCustomValidity设置“重复登录密码”表单控件的出错信息为“两次输入的密码不一致！”，否则设为空。

5.4.4 相关知识

HTML 5 CSS选择器API

W3C发布了一组通过CSS选择器来获取DOM节点的API规范，如document.

querySelector(selector)、document.querySelectorAll()。该规范的参考网站为http://www.w3.org/TR/2012/WD-selectors-api-20120628/和http://www.w3.org/TR/2012/WD-selectors-api2-20120628/。



5.5 示例5 搞定输入框自动聚焦问题

5.5.1 示例效果

本示例会演示一个HTML 5新属性autofocus。autofocus的功能是当页面加载时，元素自动获得焦点。平时可以使用这一属性增强表单的用户体验，不过使用时需小心，这个新属性在各个浏览器上的表现有一定的差异。使用Chrome浏览器打开网页文件，如图5.18所示。

再使用Firefox浏览器打开网页文件，如图5.19所示。



图5.18 使用Chrome打开网页文件



图5.19 使用Firefox打开网页文件

当一个表单内出现了多个带有autofocus属性的元素时，Chrome和Firefox会有不同的表现形式。Chrome会将焦点放到最后一个带有autofocus属性的元素，而Firefox则将焦点放到第一个带有autofocus的元素。根据W3C（World Wide Web Consortium，万维网联盟）的规定，表单内不应该出现一个以上的自动对焦元素，所以在使用的时候请务必注意这个浏览器在新标准上实现的差异，详见网址http://www.w3.org/wiki/HTML5_form_additions#autofocus。

接着使用IE 8打开文件，如图5.20所示。



图5.20 使用IE 8打开网页文件

由于IE 8不支持autofocus新属性，所以这里使用脚本模拟对应行为，采用Firefox的默认效果。



采用Firefox的效果是考虑到表单的输入顺序是至上而下的，所以应该选择最先出现autofocus的那个元素。

5.5.2 代码设计

利用编辑器打开“005.搞定输入框自动聚焦.html”文件，代码如下：

```
<!doctype html>
<html>
<head>
    <style>
        /* 省略样式，因样式与示例003相同 */
    </style>
    <script src="../js/jquery-1.8.3.js"></script>
    <script src="../js/modernizr.custom.2.6.2.js"></script>
</head>
<body>
    <header><h2>搞定输入框自动聚焦</h2></header>
    <section>
        <form action="" method="post">
            <div class="clearfix">
                <!-- 第1个autofocus -->
                <input type="email" tabindex="1" id="email"
class="item-email" placeholder="电子邮箱"
autofocus required/>
            </div>
            <div class="clearfix">
                <!-- 第2个autofocus -->
                <input type="text" tabindex="2" id="name" class="item-
name" placeholder="昵称" autofocus required/>
            </div>
            <div class="clearfix">
                <!-- 第3个autofocus -->
                <input type="password" tabindex="3" id="password"
class="item-password" placeholder="密码" autofocus autocomplete="off"
required/>
            </div>
            <div class="clearfix"><input type="submit" tabindex="4"
id="submit" value="注    册" /></div>
        </form>
    </section>
</body>
<script>
    $(document).ready(function () {          //绑定浏览器domready事件
        if (!Modernizr.input.autofocus) {
         //判断浏览器是否支持input的autofocus属性
```

```
36              $('input[autofocus]').eq(0).trigger('focus');
                //获取第一个带有autofocus属性的input并设置焦点
37          };
38      });
39  </script>
40  </html>
```



由于代码与示例003多数相同，这里只贴出差异部分，请读者注意。

5.5.3 代码分析

本示例需要引用jQuery和Modernizr两个类库，利用类库在不支持autofocus的情况下进行行为模拟。见代码第34~38行。

第34行添加浏览器的DOM加载事件监听，所有业务逻辑代码均在DOM加载完毕以后执行，代码如下：

```
$(document).ready(function(){});
```



想了解更多ready函数用法可以参考网站http://api.jquery.com/ready/。

第35行代码使用Modernizr库判断浏览器是否支持input的autofocus属性。

第36行代码在判断浏览器不支持autofocus的情况下执行。获取页面上所有带autofocus属性的input元素，并匹配第0个索引，触发focus事件设置焦点。

可以注意到表单里共有三个带autofocus属性的input元素，分别为“电子邮箱”、“昵称”和“密码”。在Firefox和IE 8中，页面加载完毕以后会自动将焦点放到“电子邮箱”，Chrome则将焦点放到“密码”文本框。

5.5.4 相关知识

1. DOMReady与window.onload区别

在刚才的示例中，所有的脚本逻辑均在DOM加载完毕以后执行。也就是说DOMReady是在所有DOM节点全部加载完成后触发，而window.onload则表示页面上的所有资源加载完成（包含图片、Flash等），时间会比DOMReady长。可惜的是DOMReady并不是浏览器原生支持的事件，不过不用担心，市面上各种类库都提供了对应的实现，如MooTools、YUI等。

2. W3C

万维网联盟，又称W3C理事会。1994年10月在麻省理工学院计算机科学实验室成立，建立者是万维网的发明者蒂姆•伯纳斯•李。



示例6 搞定表单的自动完成

5.6.1 示例效果

本示例会演示一个HTML 5新属性autocomplete，当表单页面重新加载，电子邮箱表单控件自动完成，密码表单控件不提示自动完成时使用。使用Chrome浏览器打开网页文件，单击电子邮箱输入框，如图5.21所示。

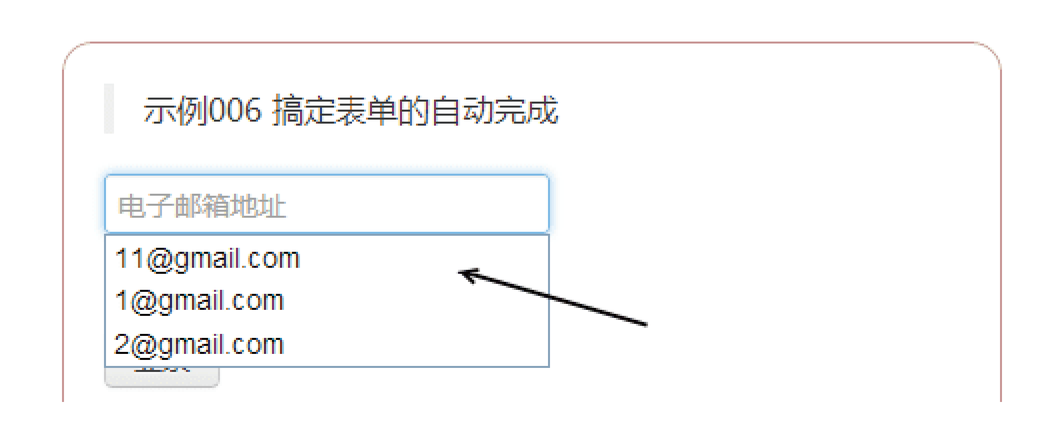


图5.21 表单自动完成下拉提示



如果读者的Chrome浏览器不支持该属性，请单击Chrome浏览器菜单，选择设置，单击“显示高级设置”选项，然后找到“密码和表单”字段，选中“启用自动填充功能后，只需单击一次即可填写多个网络表单”复选框。

5.6.2 代码设计

利用编辑器编辑如下代码，并保存为“006.搞定表单的自动完成.html”文件，代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>示例006 搞定表单的自动完成</title>
    <link rel="stylesheet" href="bootstrap.min.css" type="text/css">
    <style type="text/css">
    body{
      margin:50px auto;width:430px;padding:20px;border:1px solid #c88e8e;
      border-radius: 15px; <!-- 设置圆角-->
    }
    </style>
</head>
<body>
      <!-- 引用bootstrap的块样式 -->
```

```
16       <blockquote>
17           <p>示例006 搞定表单的自动完成</p>
18       </blockquote><!-- // 块结束 -->
19       <!-- 登录表单 -->
20         <form action="autocomplete.php" method="get"
21 autocomplete="off"><!-- 表单启用自动完成属性（全局设置） -->
22                <!-- 手动启用电子邮箱表单控件的自动完成功能 -->
23                <input type="email" name="email" id="email" value=""
24 autocomplete="on" placeholder="电子邮箱地址" />
25                <!-- 手动关闭密码表单控件的自动完成功能 -->
26                <input type="password" id="password" name="password"
   autocomplete="off" placeholder="密码" />
27                <br />
28                <input type="submit" class="btn" value="登录"/>
29         </form>
30         <!-- // 登录表单结束 -->
31 </body>
32 </html>
```

5.6.3 代码分析

HTML 5表单新增autocomplet属性。该属性适用于form标签，以及input标签类型，如text、search、url、telphone、email、password、datepickers、range及color。其中，定于form上的autocomplete属性影响其子元素的input属性。autocomplete属性的默认值为on，表示开启。

第20行设置该登录表单的autocomplete全局属性为关闭状态。

第23行打开“电子邮箱”表单控件的自动完成功能。



浏览器会记住“电子邮箱”表单的输入历史，当重新刷新该表单页面，单击输入框输入数据（或者单击上下键），会自动提示匹配的历史输入数据。

第26行设置autocomplete属性值为off，关闭“密码”表单控件的自动完成功能。



5.7 示例7　使用数字微调控件

5.7.1 示例效果

这是一个非常有趣的示例，整个页面就1个“年龄”数字文本输入框。可以通过话筒或键盘的方式进行输入，也可以通过单击文本框右侧的向上或向下箭头进行数字增减（每次增减1）。输入完毕以后，程序会根据输入的结果返回对应的出生年和生肖显示在文本框下方。



本示例没有使用任何第三方类库，代码在Chrome下测试通过，建议使用Chrome打开。

使用Chrome打开网页文件，运行效果如图5.22所示。

在“年龄”右侧文本框中输入“28”，下方会自动显示出生和生肖提示，运行效果如图5.23所示。



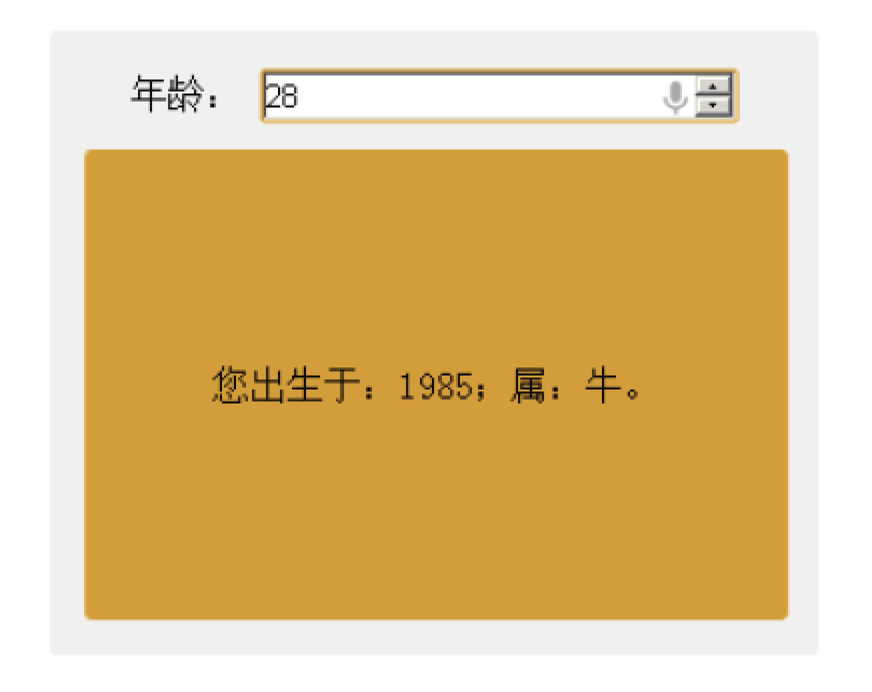




图5.22 使用Chrome打开网页文件

图5.23 文本框中输入“28”

单击文本框右侧的向上箭头，文本框内数字加1，变为“29”，同时下方提示信息发生相应的变化，运行效果如图5.24所示。单击文本框右侧的向下箭头，文本框内数字减1，还原为“28”，同时下方的提示信息也发生相应的变化。接下来单击文本框右侧话筒形状图标，运行效果如图5.25所示。



图5.24 单击向上箭头

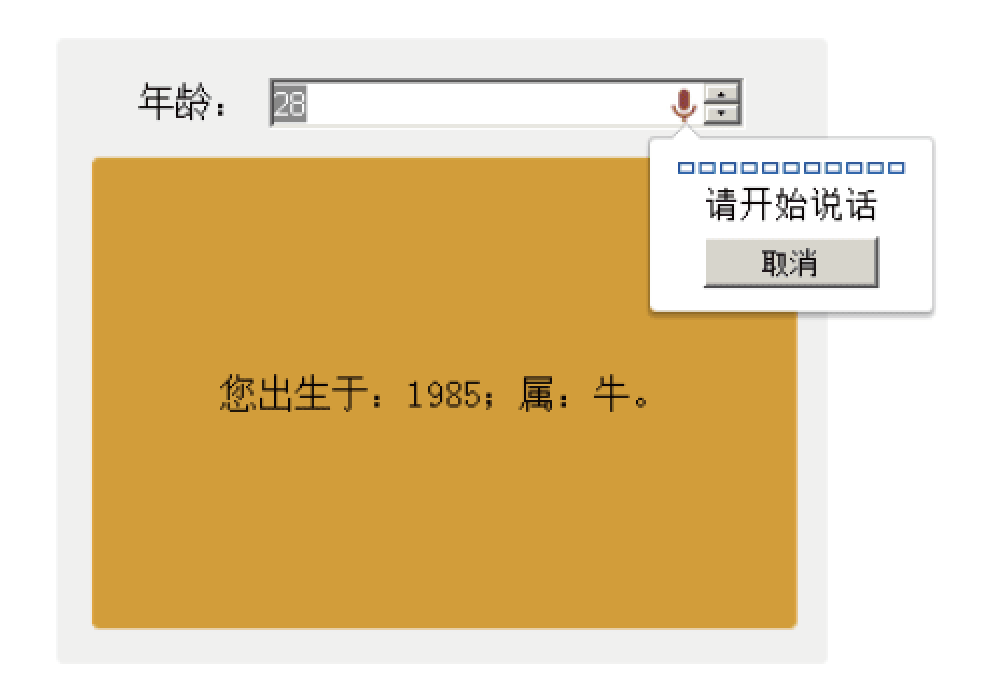


图5.25 单击话筒形状图标

如图5.25所示，出现一个带有“请开始说话”的提示框，可以对着话筒说出数字。浏览器语音识别成功以后会自动填入文本框，如果识别失败则提示“无法识别语音”，运行效果如图5.26所示。清空文本框内容，然后输入非数字型字符，如“a”，此时下方会给出错误提示，运行效果如图5.27所示。

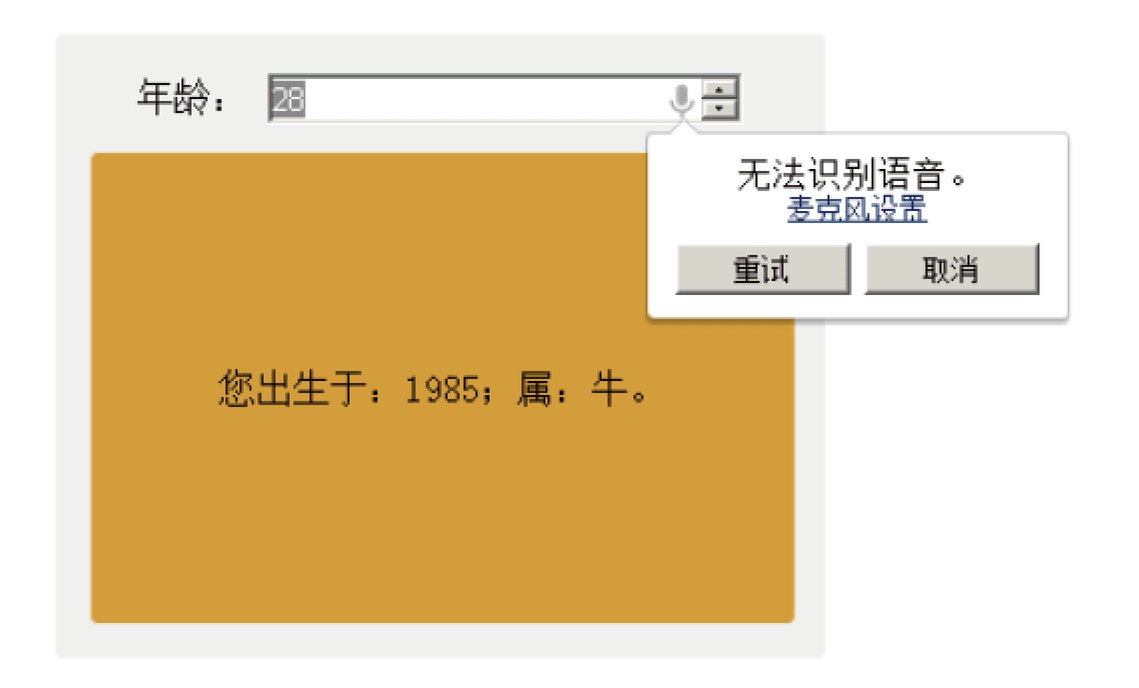


图5.26 浏览器语音识别失败

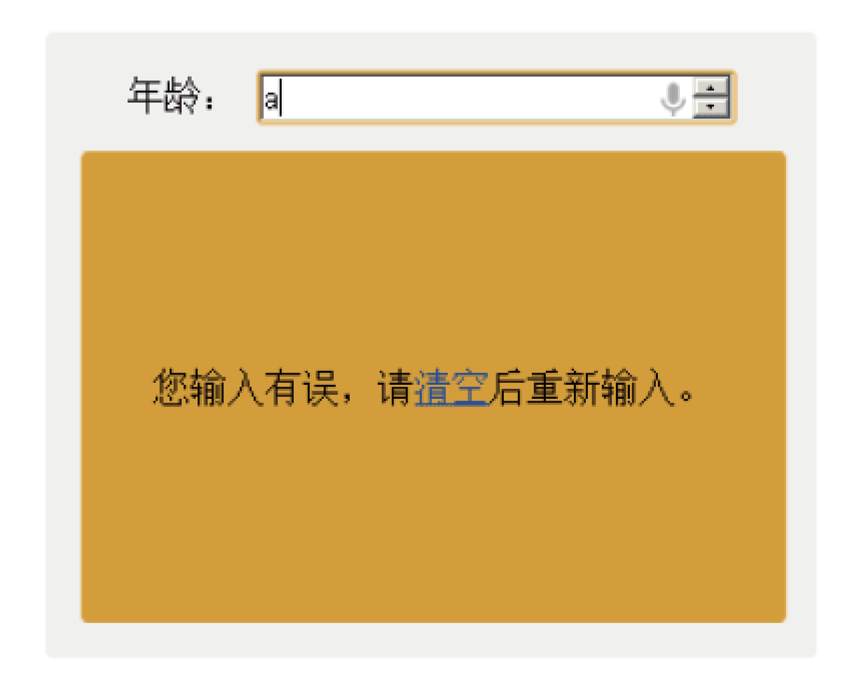


图5.27 输入非数字字符

以上就是整个示例的所有效果，下面进一步分析示例中运用了哪些HTML 5新特性和开发方面的相关知识。

5.7.2 代码设计

利用编辑器打开“007.使用数字微调控件.html”文件，代码如下：

```
<!doctype html>
<html>
<head>
    <style>
        html { text-align:center; }             /* 所有文本居中 */
        #spinner { width: 200px; }
        section
        {
            margin:0 auto;
            width:300px;
            padding:15px;
            background-color:#eee;
            border-color: #eee;
            border-radius: 3px;         /* 圆角，水平半径为3px */
            -moz-border-radius: 3px;
            -Webkit-border-radius: 3px;
        }
        .name-tip                               /* 提示框样式 */
        {
            background-color:#D5982D;
            width:300px;
            height:200px;
            line-height:200px;
            margin-top:10px;
            border-width: 1px;
            border-style: solid;
            border-color: #D69E31 #E3A037 #D5982D #E3A037;
            -moz-border-radius: 3px;    /* 圆角，水平半径为3px */
            -Webkit-border-radius: 3px;
```

```
                border-radius: 3px;
            }
        </style>
</head>
<body>
    <header><h2>使用数字微调控件</h2></header>
    <section>
         <div>
            <label for="spinner">年龄: </label>
            <span class="age">
                <!-- speech x-Webkit-speech 表示允许用语音输入-->
                <input id="spinner" name="spinner" type="number"
min="1" max="100" step="1" required speech x-Webkit-speech />
            </span>
        </div>
        <!-- 提示容器 -->
        <div class="name-tip" id="J_name_tip"></div>
    </section>
</body>
<script>
    var spinner = document.getElementById('spinner'),
    // 获取数值文本框元素
        name_tip = document.getElementById('J_name_tip'),
        // 获取提示文本框容器
        arr = ['猴', '鸡', '狗', '猪', '鼠', '牛', '虎', '兔', '龙', '蛇',
 '马', '羊']; // 生肖数组

    function getTip() {
        var number = parseInt(spinner.value),
        // 转换为整型
            year = (new Date().getFullYear() - number),
            // 获取出生年
            msg;

        if (!spinner.validity.valid) {
        // 判断是否为数字类型
                   // 填充错误信息，并添加“清空”功能
            msg = '您输入有误，请<a href="javascript:spinner.value=\'\'">
清空</a>后重新输入。';
        } else {
                   // 用正则表达式判断是否为4位数字，如果是则取数组索引是12的模
            msg = '您出生于: ' + year + '; 属: ' + (/^\d{4}$/.test(year)
 ? arr[year % 12] : '未知') + '。';
        };
        name_tip.innerHTML = msg;
        // 填充提示容器HTML信息
    };
    spinner.addEventListener('input', getTip, false);
    // 数值文本框input事件
    spinner.addEventListener('speechchange', getTip, false);
    // 数值文本框speechchange事件
```

```
70 </script>
71 </html>
```



本示例代码在Chrome下测试通过，请选用Chrome打开示例浏览。

5.7.3 代码分析

代码第5行~第31行设置了文本框和提示框的样式，其中用到了CSS 3的border-radius（示例003中已经有非常多的讲解，这里不多加赘述）。

代码第41行~第42行，上面集合了多个HTML 5 input新属性，有min、max、step、required、speech（x-Webkit-speech），代码如下：

```
<input id="spinner" name="spinner" type="number" min="1" max="100"
 step="1" required speech x-webkit-speech />
```

其中HTML 5相关新属性的说明如下。

- min：输入数字的最小值。
- max：输入数字的最大值。
- step：输入数字的合法数字间隔。
- required：输入字段的值是必需的。
- speech：打开语音输入功能。



x-webkit-speech是WebKit核心的浏览器的私有属性，比较常见的WebKit核心浏览器有Chrome、国内双核浏览器最新版（如傲游、360等）。如果想了解更多的W3C语音输入API规范可以参考网站http://www.w3.org/2005/Incubator/htmlspeech/。

代码第50行~第69行是主要的脚本代码，接着看看这些代码到底做了些什么。

第50行~第52行定义了三个变量，前两个变量获取DOM中数字文本框和提示框元素，第三个变量定义了一个生肖数组用于后面的信息提示。

第54行~第67行定义了一个函数，用于数值文本框的事件绑定。第59行运用了HTML 5表单的一个新对象ValidityState，即spinner.validity获取的对象值，该对象拥有8个属性，说明如下。

- valueMissing：布尔型，必填表单元素的值为空时为true，否则为false。
- typeMismatch：布尔型，输入值与type类型不匹配时为true，否则为false。
- patternMismatch：布尔型，输入值与pattern属性正则不匹配时为true，否则为false。
- tooLong：布尔型，输入内容超出元素maxLength属性数值时为true，否则为false。
- rangeUnderflow：布尔型，输入值小于min属性数值时为true，否则为false。
- rangeOverflow：布尔型，输入值大于max属性数值时为true，否则为false。
- stepMismatch：布尔型，输入值不匹配step特性规则时为true，否则为false。
- customError：布尔型，是用户自定义错误信息时为true，否则为false。

第64行代码当输入符合要求时触发，同时通过正则和模运算获取年对应的生肖数值，关

键逻辑代码如下：

```
/^\d{4}$/.test(year) ? arr[year % 12] : '未知'
```

第68行~第69行代码绑定文本框的input和speechchange事件，分别在用户输入结束和语音输入结束时触发，代码如下：

```
spinner.addEventListener('input', getTip, false);
spinner.addEventListener('speechchange', getTip, false);
```

addEventListener方法共有三个参数，分别表示事件名称、事件函数名、是否使用捕获。

HTML 5事件类型远远不止示例中的这些，想了解更多HTML 5事件属性请参考网站http://www.w3school.com.cn/html5/html5_ref_eventattributes.asp。

5.7.4 相关知识

1. 正则表达式

正则表达式的英文名为Regular Expression，是计算机科学的一个概念。正则表达式使用单个字符串来描述、匹配一系列符合某个句法规则的字符串。JavaScript的正则表达式有两种写法，如下：

```
/pattern/attributes                    // 直接量语法
new RegExp(pattern, attributes);       // 创建 RegExp 对象的语法
```



想了解更多的JavaScript正则表达式请前往http://www.w3school.com.cn/js/jsref_obj_regexp.asp。

2. DOM事件流

DOM事件流分为两种：捕获型和冒泡型。捕获型事件只在非IE浏览器下得到支持，不过冒泡型事件在所有浏览器下都得到了支持。在刚刚的示例中用到了冒泡型事件，给一张图看看冒泡型事件是怎么工作的，如图5.28所示。图片清楚地说明了冒泡型事件是一个由内到外的过程。事件从目标元素开始，通过各个父节点穿过整个DOM。接着通过图示看看捕获型事件流的过程，如图5.29所示。捕获型事件是一个由外到内的过程，从DOM的根元素开始，依次由父元素逐级向下到达目标元素。

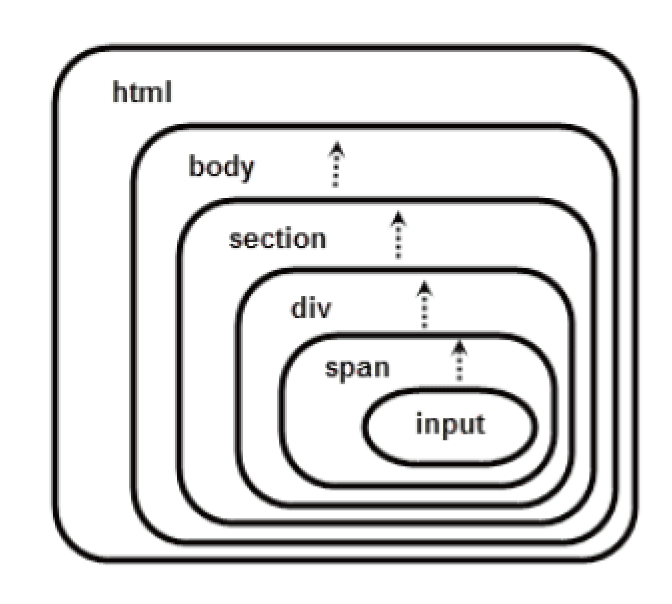


图5.28 冒泡型事件流

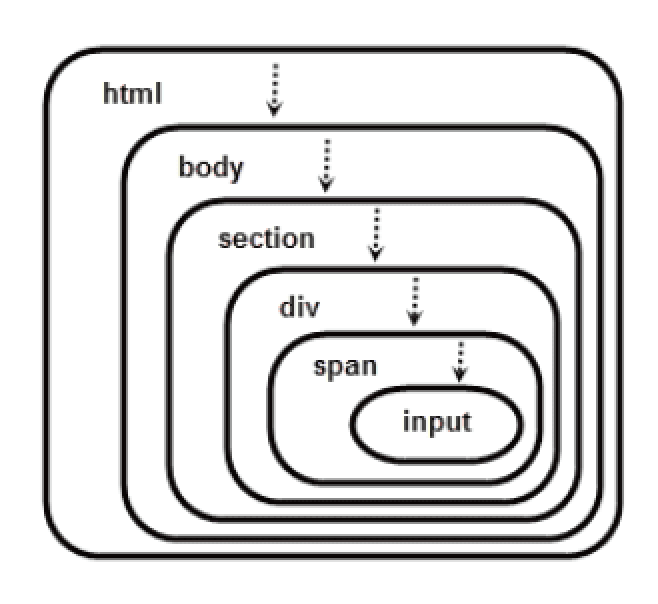


图5.29 捕获型事件流



了解更多知识可以参考网站http://www.quirksmode.org/js/events_order.html。



5.8 示例8 添加滑动控件

5.8.1 示例效果

通常一个带有抽奖性质的后台活动管理系统，都可以灵活地设置中奖概率。本示例演示了一个通过HTML 5的滑动控件控制转盘抽奖系统的中奖百分比的示例。可以通过滑动条滑动设置中奖概率，滑动条滑动时，右侧将显示当前数值。



代码在Chrome、Safari、Opera下测试通过，建议使用Opera打开。

使用Opera浏览器打开网页文件，运行效果如图5.30所示。

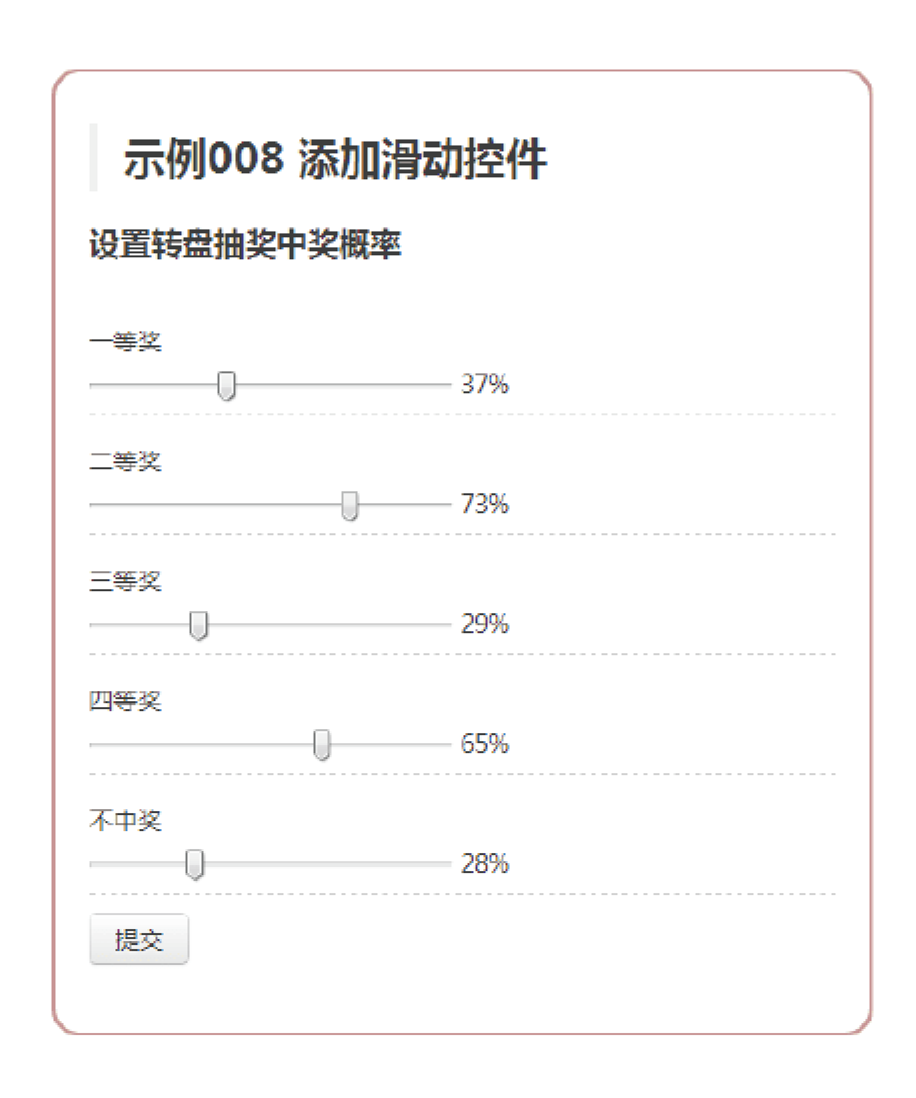


图5.30 设置中奖概率

5.8.2 代码设计

利用编辑器打开“008 添加滑动控件html”文件，代码如下：

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>示例008 添加滑动控件</title>
    <link rel="stylesheet" href="bootstrap.min.css" type="text/css">
      <script src="jquery-1.8.3.js"></script>
    <style type="text/css">
    body{
      margin:50px auto;width:430px;padding:20px;border:1px solid #c88e8e;
      border-radius: 15px;                              <!-- 设置圆角-->
    }
      .control-group{border-bottom:1px dashed #ccc;padding: 5px 0;}
    </style>
</head>
<body>
      <!-- 引用bootstrap的块样式 -->
    <blockquote>
        <h3>示例008 添加滑动控件</h3>
    </blockquote>                                       <!-- 块结束 -->
    <!-- 中奖概率设置 -->
      <h4>设置转盘抽奖中奖概率</h4>
      <br>
      <form action="#" method="get" autocomplete="on">
             <div class="control-group">
                    <label for="one">一等奖</label>
                    <!-- 中奖概率设置滑动控件 -->
                    <input type="range" class="range" name="one" id="one" min="0" max="100" step="1" />
                    <!-- 显示控件 -->
                    <output onforminput="value=one.value + '%'">50%</output>
             </div>
             <div class="control-group">
                    <label for="two">二等奖</label>
                    <input type="range" class="range" name="two" id="two" min="0" max="100" step="1" />
                    <output onforminput="value=two.value + '%'">50%</output>
             </div>
             <!-- ... 其他奖项表单控件同上 ...-->
             <input type="submit" class="btn" value="提交"/>
      </form>
      <!-- // 表单结束 -->
</body>
<script>
      $(function(){
             var ranges = $(".range"), i;
             for(i = 0, max = ranges.length; i < max; i += 1){
                    var that = $(ranges[i]);
                    var bind = (function(t){
```

```
                    // 把变量放到闭包里
47                          var output = t.next('output');
                            // 获取显示控件
48                          var showValue = function(v){
                            // 降级处理，显示值
49                                output.text(v + '%');
50                          }
51                          t.bind("change", function(){
                            // 绑定滑动控件的change事件
52                                val = t.val();
53                                showValue(val);
54                          });
55                    }(that));
56              }
57        });
58 </script>
59 </html>
```

5.8.3 代码分析

第28行、第34行设置中奖概率滑动控件的类型和属性。类型为range表示该控件是一个滑动控件，滑动控件有三种属性。

- min：滑动的最小值，当滑动操作滑到最左边时，控件值最小，在本例中为0。
- max：滑动的最大值，当滑动操作滑到最右边时，控件值最大，在本例中为100。
- step：滑动增量，滑动操作的增量值，在本例中为1，表示滑动值处理0~100间的自然数，不会出现0.1、1.1这样的数值。

第30行、第35行设置滑动值显示控件，当滑动操作发生时，当前值会显示在该控件下。



只有Opera浏览器支持output与range相关联。

代码第42行~第56行处理非Opera浏览器显示控件值的降级。

第43行，获取所有滑动控件。
第44行，循环所有滑动控件。
第46行，用闭包保护变量，存储当前的滑动控件及值显示控件。
第48行，当前滑动值显示函数。
第51行，绑定滑动控件的change事件，如果控件当前值改动，则更新显示控件的值。

5.8.4 相关知识

output控件

在Web站点上会经常看到一些用于计算贷款利率、税收、保险以及更多事情的控件等（比如计算器网站）。output元素用来标记这些计算的结果，如以下表单代码所示：

```
<form onsubmit="return false" oninput="o.value = parseInt(a.value)
  + parseInt(b.value)">
  <input name="a" type="number" step="any"> +
  <input name="b" type="number" step="any"> =
  <output name="o"></output>
</form>
```



想要了解更多关于output标记的值，请参考网站http://html5doctor.com/the-output-element/。



示例9 发送多个文件

5.9.1 示例效果

在传统的文件上传中，每次只允许上传一个文件。HTML 5新规范在input上添加了多文件上传属性，每次允许上传多个不同文件。本示例分别给出了客户端和服务器端的实现代码。通过本示例，可以学习使用input的新属性multiple，同时了解服务器端的存储。

使用Chrome浏览器打开网页文件，运行效果如图5.31所示。



图5.31 使用Chrome打开网页文件

页面共由两部分组成：表单和上传文件列表。表单中包含了两个input，类型分别为file和submit。单击“选择文件”按钮，选中“图片库”文件夹中的多个文件，如图5.32所示。



图5.32 选择多个上传文件

单击“打开”按钮，选中的文件信息显示在下方列表中，如图5.33所示。

选择文件 8 个文件 提交

文件名	类型	大小
Chrysanthemum.jpg	image/jpeg	879394
Desert.jpg	image/jpeg	845941
Hydrangeas.jpg	image/jpeg	595284
Jellyfish.jpg	image/jpeg	775702
Koala.jpg	image/jpeg	780831
Lighthouse.jpg	image/jpeg	561276
Penguins.jpg	image/jpeg	777835
Tulips.jpg	image/jpeg	620888

图5.33 文件信息显示列表

文件信息列表中列出了“文件名”、“类型”和“大小”三项数据。在单击“提交”按钮之前先启动Node.js上传服务器。进入“009.upload-server”文件夹，如图5.34所示。



图5.34 “009.upload-server”文件夹内容

server.js存放了主要的上传服务逻辑脚本，package.json存放了模块的描述信息，node_modules文件夹存放程序依赖模块。在Windows下打开命令行进入刚才的目录，执行如下代码：

```
node server.js
```

如果启动成功，命令行提示如下：

```
listening on http://localhost:8080
```

返回到如图5.33所示的状态，单击“提交”按钮，进行文件上传。上传成功后会提示“上传成功，文件已经保存到服务器，远程路径：xxx”。同时命令行会显示相应上传的文件名称，如图5.35所示。

```
listening on http://localhost:8080
Chrysanthemum.jpg
Desert.jpg
Hydrangeas.jpg
Koala.jpg
Lighthouse.jpg
Penguins.jpg
Tulips.jpg
Jellyfish.jpg
```

图5.35 文件上传成功提示

至此整个上传过程结束。快来亲手体验一下多文件上传的魅力吧！



如果服务器在本机启动，这里的“路径：xxx”就是对应的计算机临时文件夹路径，如“C:\Users\(loginName) \AppData\Local\Temp”。

5.9.2 代码设计

利用编辑器打开“009.发送多个文件.html”文件，代码如下：

```
<!doctype html>
<html>
<head>
    <style>
        /* 示例样式，具体查看源代码文件 */
        </style>
      </head>
      <body>
          <header><h2>发送多个文件</h2></header>
          <section>
              <!-- 表单编码类型为“multipart/form-data”，post到本机服务器8080端口 -->
              <form method="post" enctype="multipart/form-data" action="http://localhost:8080" >
                  <!-- 多文件上传控件，增加了multiple属性 -->
                  <input type="file" name="multi-file" id="multi-file" multiple />
                  <input type="submit" value="提交" />
              </form>
              <!-- 文件显示列表 -->
              <table id="file-list"><tr><td>空</td></tr></table>
          </section>
```

```
    </body>
    <script>
        var multi_file = document.getElementById('multi-file'),
        // 获取上传控件
            file_list = document.getElementById('file-list'),
            // 获取列表控件
            slice = Array.prototype.slice;
            // 获取数组slice原型方法
        multi_file.addEventListener('change', function (e) {
        // 监听上传控件数据变化
            var files = slice.call(multi_file.files, 0),
            // 将FileList对象转为数组
                htmls = ['<tr><th>文件名</th><th>类型</th><th>大小
</th></tr>'];        // 列表标题
            files.forEach(function (file, index) {
            // 循环file数组
                // 组装 File对象的 name、type、size属性为html
                htmls.push('<tr><td>' + file.name + '</td><td>'
+ file.type + '</td><td>' + file.size +
    '</td></tr>');
            });
            file_list.innerHTML = htmls.join('');
            // 重设列表html
        }, false);
    </script>
</html>
```

利用编辑器打开“009.upload-server”文件夹中的server.js文件，代码如下：

```
var formidable = require("formidable"),
// 引用formidable模块，用于文件保存
    fs = require('fs'),
    // 引用fs模块，用于对文件进行操作
    http = require("http");
    // 引用http模块，用于web服务器

http.createServer(function (req, res) {              // 创建新服务器
    var form = new formidable.IncomingForm(), // 新建form实例
        files = [];
        // 新建files数组，用于存储上传文件实例

    form.on('end', function () {
    // 监听form上传结束事件
        // 设置响应头MIME类型和编码
        res.setHeader('content-type', 'text/html;charset=utf-8;');
        // 写入上传成功信息
        res.write('上传成功，文件已经保存到服务器，远程路径：' +
form.uploadDir);
        // 响应结束
        res.end();
    }).on('file', function (field, file) {
```

```
        // 监听form文件上传事件
17          files.push(file);
            // 将上传文件的file实例存入数组
18      }).on('error', function (error) {
        // 监听form上传出错事件
19          res.setHeader('content-type', 'text/html;charset=utf-8;');
20          res.write('上传失败，' + error);
21          res.end();
22      });
23      form.parse(req, function (err) {
        // 调用parse接口解析请求信息并保存文件
24          files.forEach(function (item, index) {
            // 循环上传的文件数组
25              console.log(item.name);
                // 在控制台显示文件名
26              // 将上传至服务器端的文件名重命名为原文件名
27              fs.rename(item.path, form.uploadDir + '\\' + item.name,
    function (err) { });
28          });
29      });
30  }).listen(8080, function () {
    // 设置web服务器监听端口，并启动服务
31      // 控制台显示web服务器启动成功
32      console.log('listening on http://localhost:8080');
33  });
```

5.9.3 代码分析

1. 客户端网页代码分析

代码第12行中form表单的enctype属性值为“multipart/form-data”，表示传输不对字符编码，进行二进制传输，当表单有上传控件时，需要加上这个参数。action地址设置为本机的8080端口，指向上传服务器。method为post，表示表单以post方式传输。

代码第14行为上传控件，添加input元素为HTML 5新属性multiple，表示同时可以添加多文件上传。

代码第24行获取了Array原型上的slice方法。slice方法返回一个新数组，包含从原数组中选定的元素，slice语法如下：

```
arrayObject.slice(start,end)
```

第25行~第34行在上传控件上监听change事件，当控件数值变化时触发。

第26行代码，multi_file.files获取到一个FileList对象。该对象拥有一个只读的length属性表示列表长度。同时对象上拥有item方法，通过该方法获取对应索引的数据，item方法语法如下：

```
filelistObject.item(index)
```

这里将FileList对象通过slice方法转化为数组，从而获取数组的forEach方法。

代码第28行~第32行循环获取文件列表中的File对象，将对象上的name（文件名）、type

（文件类型）、size（文件大小）属性拼接为字符串填入htmls数组。

代码第33行调用数组join方法将htmls数组放入一个字符串，然后填充到文件列表元素中，至此核心的客户端代码业务逻辑完毕。

2．服务器端代码分析

通过第4章的学习，已经了解到Node.js基本的使用方法，本示例运用Node.js搭建一个上传服务器。下面来看看服务器端是如何实现的。

代码第1行~第3行引用了本次需要使用的三个模块。

- formidable：解析表单传送数据，用于处理文件上传。
- fs：文件系统模块，用于处理文件重命名、读取、写入等。
- http：HTTP模块，用于处理发送请求、接受请求等。



formidable模块的更多使用方法可以参考https://github.com/felixge/node-formidable，fs和http的使用可以参考http://docs.cnodejs.net/cman/index.html。

代码第5行，调用http的createServer方法创建一个HTTP服务器，同时传入一个函数。当有请求时会触发传入函数，函数获得请求对象和响应对象（即代码中的req和res参数）。

代码第6行新建了一个formidable模块的IncomingForm类实例，该类是对提交表单的抽象。

第9、16、18行代码分别监听了IncomingForm实例的end、file和error事件。

代码第11行和第19行设置了响应头信息，将“content-type”设置为“text/html;charset=utf-8”，表示响应信息为互联网媒体类型，编码格式为UTF-8。



text/html类型更多信息可以参考http://tools.ietf.org/html/rfc2854。

代码第13行往响应数据流中写入提示信息，接着在代码第15行调用end方法，表示所有数据都已经被发送，响应可以结束（代码第20和21行与此相同）。

代码第17行监听提交表单的上传事件，获得的文件对象推入临时创建的数组，在代码第24行会对该数组进行循环操作。

代码第27行接收循环文件数组并获取单个文件对象，调用代码第二行引用的fs模块（文件系统模块），通过fs模块的rename方法修改对应上传至服务器端的文件名，代码如下：

```
fs.rename(item.path, form.uploadDir + '\\' + item.name, function (err) { });
```

代码第30行启动服务并监听8080端口，在回调函数中打印启动成功信息。



示例多处使用Node.js的HTTP模块，这个模块在Web服务器开发中至关重要，想更深入地了解相关API使用可以参考http://nodejs.org/api/http.html#http_response_end_data_encoding。

5.9.4 相关知识

MIME编码

例子中多次出现“content-type”，在HTTP当中MIME类型被定义在其中。MIME类型是

多用途互联网邮件扩展的一个互联网标准，英文名为Multipurpose Internet Mail Extensions。最早应用在邮件系统当中，后来应用到浏览器中。常见的类型有text/html（HTML文档）、text/plain（纯文本）、image/gif（GIF图像）、image/jpeg（JPEG图像）等。更多信息可以参考http://www.w3school.com.cn/media/media_mimeref.asp和http://zh.wikipedia.org/wiki/MIME。



5.10 示例10 利用正则表达式创建自定义输入类型

5.10.1 示例效果

在一个电子商务网站中，通常需要维护用户的个人基本信息，如昵称、身份证号码等。本示例演示了在电子商务网站个人中心的基本信息设置页面，设置个人基本信息的表单。当用户单击“提交”按钮时，昵称和身份证号必须匹配表单控件的正则表达式，否则给出提醒。



代码在Chrome下测试通过，建议使用Chrome打开。

使用Chrome浏览器打开网页文件，运行效果如图5.36所示。



图5.36 利用正则表达式创建两个自定义类型

5.10.2 代码设计

利用编辑器打开“010.利用正则表达式创建自定义输入类型.html”文件，代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="bootstrap.min.css" type="text/css">
    <title>示例010 利用正则表达式创建自定义输入类型</title>
    <style type="text/css">
    body{
      margin:50px auto;width:430px;padding:20px;border:1px solid
 #c88e8e;
      border-radius: 15px;                              /*  设置圆角 */
     }
    span.msg{font-size: 10px;display: block;}         /* 验证提示信息 */
    .info{line-height: 30px; font-weight: bold;}      /* 不能填字段 */
    </style>
</head>
<body>
    <blockquote>
        <p>示例010 利用正则表达式创建自定义输入类型</p>
    </blockquote><!-- 块结束 -->
    <!-- 表单开始 -->
    <h4>在电子商务网站中设置个人基本信息</h4>
    <br>
    <form class="form-horizontal" name="myform">
        <div class="control-group">           <!-- 用户名显示区 -->
            <label class="control-label">用户名</label>
            <div class="controls info">spricity</div>
        </div>
        <div class="control-group">           <!-- 用户类型显示区 -->
            <label class="control-label">会员类型</label>
            <div class="controls info">个人用户</div>
        </div>
        <div class="control-group">
            <label class="control-label" for="nickname">昵称</label>
<!-- 昵称填写区域 -->
            <div class="controls">
                <!-- 自定义昵称输入类型 -->
                <input type="text" id="nickname" required pattern="^[0-
9A-Za-z\u4e00-\u9fa5_\-]+$" name="nickname">
                <!-- 昵称验证类型提醒 -->
                <span class="msg">可由中英文、数字、“_”、“-”组成</span>
            </div>
        </div>
        <div class="control-group">           <!-- 邮箱填写区域 -->
            <label class="control-label">邮箱</label>
            <div class="controls info">spricity@gmail.com</div>
```

```
45          </div>
46          <div class="control-group">           <!-- 手机号填写区域 -->
47              <label class="control-label">手机号</label>
48              <div class="controls info">186*****581</div>
49          </div>
50          <div class="control-group">           <!-- 身份证号填写区域 -->
51              <label class="control-label" for="cardno">身份证号</label>
52              <div class="controls">
53                  <!-- 自定义身份证号输入类型 -->
54                  <input type="text" id="cardno" pattern="^(\d{15}$|^
   \d{18}$|^\d{17}(\d|X|x))$" name="cardno">
55              </div>
56          </div>
57          <div class="control-group">           <!-- 表单提交按钮区 -->
58              <div class="controls">
59                  <button type="submit" class="btn">提交</button>
60              </div>
61          </div>
62      </form>                                   <!-- 表单结束 -->
63  </body>
64  </html>代码分析
```

第36行设置昵称控件的pattern属性，设置其属性的正则表达式的值为“^[0-9A-Za-z\u4e00-\u9fa5_\-]+$”，要求该表单控件值必须满足由中英文、数字、“_”、“-”组成。



本章第4个例子讲到的patternMismatch控件属性，在HTML 5代码上，只要设置pattern为正则表达式的值，就可以使用表单控件的patternMismatch属性。

第54行，定义身份证号控件，设置其pattern值为“^(\d{15}$|^\d{18}$|^\d{17}(\d|X|x))$”，该正则表达式满足身份证15位数字的身份证号，以及18位数字或者17位数字加一位字母X的身份证号码。



如果要自定义其他类型的表单控件，只要在pattern属性上定义相应的正则表达式即可。



5.11 示例11 预览上传的图片

5.11.1 示例效果

在示例9中，介绍了HTML 5新特性multiple和上传服务器端的存储逻辑脚本。本示例将在原有的基础上增加上传类型判断和图片预览功能，增强表单的用户体验，同时还会介绍HTML 5在文件数据读取方面的操作。

使用Chrome浏览器打开网页文件，运行效果如图5.37所示。



图5.37 使用Chrome打开网页文件

单击“选择文件”按钮，选择“图片库”中的图片文件和两个非图片文件，如图5.38所示。

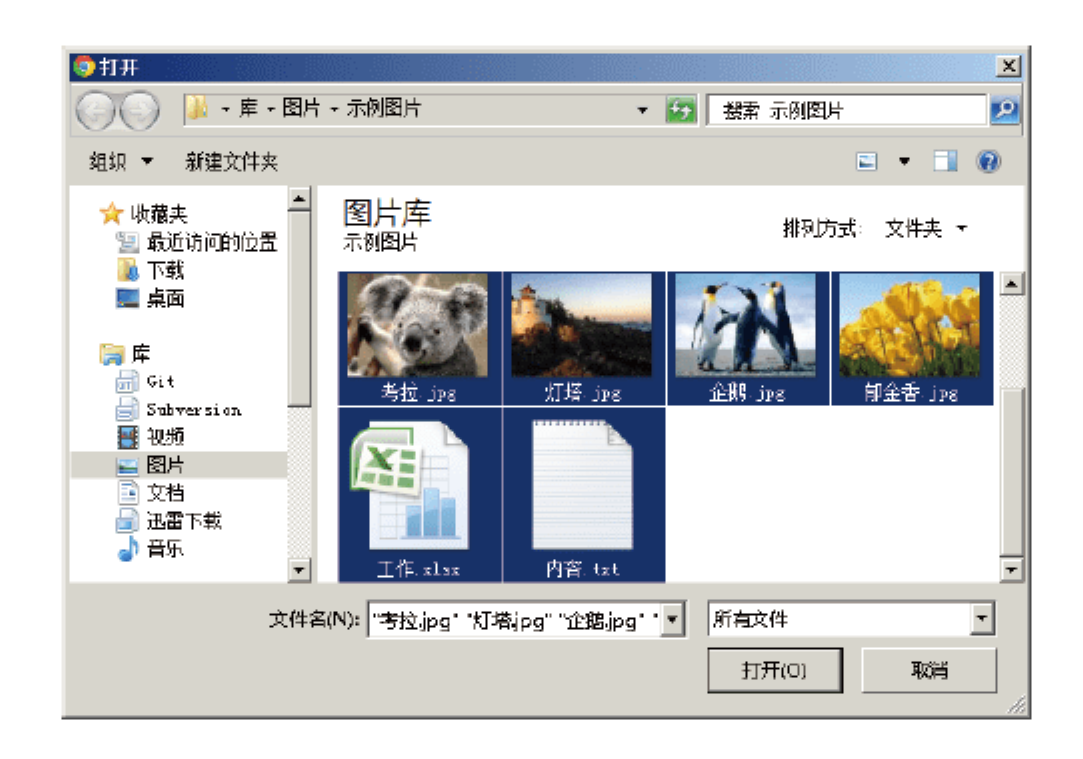


图5.38 选择图片文件和两个非图片文件

单击“打开”按钮，下方生成上传文件列表并显示图片预览。如果选中的文件为图片类型，则在预览框内展现缩略图片，否则显示为空，如图5.39所示。



文件名	类型	大小	预览
Koala.jpg	image/jpeg	780831	
Lighthouse.jpg	image/jpeg	561276	
Penguins.jpg	image/jpeg	777835	
Tulips.jpg	image/jpeg	620888	
工作.xlsx	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	9879	
内容.txt	text/plain	0	

图5.39 上传文件列表和图片预览



后续操作与示例009相同，这里不过多介绍。

5.11.2 代码设计

利用编辑器打开“011.预览上传的图片.html”文件，代码如下：

```
<!doctype html>
<html>
<head>
    <style>
        /* 与示例009基本相同，略... */
    </style>
</head>
<body>
    <header><h2>预览上传的图片</h2></header>
    <section>
        <!-- 表单编码类型为“multipart/form-data”，post到本机服务器8080端口 -->
        <form method="post" enctype="multipart/form-data"
 action="http://localhost:8080" >
            <!-- 多文件上传控件，增加了multiple属性 -->
            <input type="file" name="multi-file" id="multi-file" multiple />
            <input type="submit" value="提交" />
        </form>
        <!-- 文件显示列表 -->
        <table id="file-list"><tr><td>空</td></tr></table>
    </section>
</body>
<script>
(function () {
    var multi_file = document.getElementById('multi-file'), // 获取上传控件
        file_list = document.getElementById('file-list'), // 获取列表控件
        slice = Array.prototype.slice;        // 获取数组slice原型方法
    var GUID = 0;
    function guid() { return GUID++; };       // 全球唯一标识符生成函数
    multi_file.addEventListener('change', function (e) {
    // 监听上传控件数据变化
        var files = slice.call(multi_file.files, 0),
        // 将FileList对象转为数组获取forEach方法
            htmls = ['<tr><th>文件名</th><th>类型</th><th>大小</th><th>
预览</th></tr>'];   // 列表标题

        files.forEach(function (file, index) {  // 循环File数组
            var reader,         // reader变量用于存放FileReader实例
                _guid = guid();               // 获取全球唯一标识符
            // 组装 File对象的 name、type、size属性为html
            htmls.push('<tr><td>' + file.name + '</td><td>' + file.type
+ '</td><td>' + file.size + '</td><td id="J_' + _guid + '"></td></tr>');
            if (file.type.toLowerCase().match(/image.*/)) {
```

```
            //用正则表达式判断文件是否为图片类型
39                  reader = new FileReader();
                    // 实例化FileReader对象，用于读取文件数据
40                  reader.addEventListener('load', function (e) {
                    // 监听FileReader实例的load事件
41                      // 获取对应元素，并填充相应图片
42                      document.getElementById('J_' + _guid).innerHTML =
43  '<img title="' + file.name + '" alt="' 43      + file.name + '" src="'
    + e.target.result + '"/>';
44                  });
45                  reader.readAsDataURL(file);           // 读取文件为DataURL
46              };
47          });
48          file_list.innerHTML = htmls.join('');         // 重设列表html
49      }, false);
50  })();
51  </script>
52  </html>
```



服务器端脚本与示例9相同，本示例不过多说明。

5.11.3 代码分析

代码第26行，定义全局唯一标识符变量，用于保存预览框唯一标识。

代码第27行，定义函数guid，用于产生唯一标识符，每次执行后计数加1。

代码第37行，给图片预览框添加id，规则为“J_”加上唯一标识符，以免元素id重复。

代码第38行将文件类型名转化为小写字符串，然后用正则表达式匹配image字符串。如果匹配成功，表示该文件为图片类型。

代码第39行，实例化FileReader对象。FileReader拥有以下几个常用方法。

- readAsArrayBuffer：将文件读取为ArrayBuffer对象。
- readAsBinaryString：将文件读取为二进制字符串。
- readAsDataURL：将文件读取为Base64编码数据。
- readAsText：将文件读取为文本。
- abort：停止读取文件。



想了解更多FileReader接口信息可以参考http://www.w3.org/TR/file-upload/#dfn-filereader。

代码第40行，监听FileReader实例的load事件。FileReader的事件模型如下。

- onabort：读取文件中断时触发。
- onerror：读取文件出错时触发。
- onload：读取文件成功完成时触发。
- onloadend：读取文件结束时触发。
- onloadstart：读取文件开始时触发。

- onprogress：读取文件中触发。

代码第42行、第43行，在文件读取成功结束后执行，找到对应的单元格元素，将拼接的图片HTML结构填充至单元格。其中，e.target.result为对应文件获取的Base64编码数据。

5.11.4 相关知识

Base64编码是一种基于64个可打印字符来表示二进制数据的表示方法。现代的浏览器都已经原生支持Base64。可以用在如以上示例的环境，通过浏览器的API读取图片文件的Base64编码然后设置图片，还可以用在将Canvas另存为图片文件时。并且这也是前端优化的一项重要技巧，通过Base64处理图片，达到减少HTTP请求数的优化效果，俗称Data URI scheme，更多信息来自http://zh.wikipedia.org/zh-cn/Base64和http://en.wikipedia.org/wiki/Data_URI_scheme。



5.12 示例12 无刷新异步上传

5.12.1 示例效果

本示例在示例11的基础上进一步加强了用户体验。用户可以通过不刷新页面上传多个文件，并同时显示客户端上传进度。

首先启动上传Web服务器。使用Windows命令行进入“012.upload-server”文件夹，执行命令如下：

```
node server.js
```

如果启动成功，命令行提示如下：

```
listening on http://localhost:8080
```

使用Chrome浏览器打开网页文件，运行效果如图5.40所示。



图5.40 使用Chrome打开网页文件

单击“选择文件”按钮，选择“图片库”中的图片文件和两个非图片文件，如图5.41所示。

图5.41 选择图片文件和两个非图片文件

单击“打开”按钮，下方生成上传文件列表并显示图片预览和上传进度框。如果选中的文件为图片类型，则在预览框内展现缩略图片，否则显示为空，如图5.42所示。

选择文件 6 个文件　提交

文件名	类型	大小	预览	上传进度
Koala.jpg	image/jpeg	780831		
Lighthouse.jpg	image/jpeg	561276		
Penguins.jpg	image/jpeg	777835		
Tulips.jpg	image/jpeg	620888		
工作.xlsx	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	9879		
内容.txt	text/plain	0		

图5.42 上传文件列表和图片预览，增加上传进度列表

单击“提交”按钮，文件被依次上传到服务器，上传进度条同时产生变化直到上传结束，如图5.43所示。



文件名	类型	大小	预览	上传进度
Koala.jpg	image/jpeg	780831		100%
Lighthouse.jpg	image/jpeg	561276		100%
Penguins.jpg	image/jpeg	777835		100%
Tulips.jpg	image/jpeg	620888		100%
工作.xlsx	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	9879		100%
内容.txt	text/plain	0		100%

图5.43 单击“提交”按钮上传文件

5.12.2 代码设计

利用编辑器打开“012.无刷新异步上传.html”文件，代码如下：

```
<!doctype html>
<html>
<head>
    <style>
        /* 与示例009基本相同，略... */
        input[type="file"]                      /* 上传控件样式 */
        {
            width: 300px;
        }
        .progressbar                            /* 进度条外框样式 */
        {
            border: 1px solid #dddddd;
            border-radius: 4px;                 /* 圆角，水平半径为4px */
            -moz-border-radius: 4px;
            -webkit-border-radius: 4px;
            width: 80px;
            height: 14px;
        }
        .progressbar-value                      /* 进度条数值样式 */
        {
            margin: -1px;
            height: 100%;
            border: 1px solid #e78f08;
            border-radius: 4px;                 /* 圆角，水平半径为4px */
```

```
                -moz-border-radius: 4px;
                -webkit-border-radius: 4px;
                background-color: #f6a828;
            }
    </style>
</head>
<body>
    <header><h2>无刷新异步上传</h2></header>
    <section>
      <!-- 表单编码类型为“multipart/form-data”，post到本机服务器8080端口 -->
        <form method="post" enctype="multipart/form-data"
 action="http://localhost:8080" >
            <!-- 多文件上传控件，增加了multiple属性 -->
            <input type="file" name="multi-file" multiple />
            <input type="submit" value="提交" />
        </form>
        <!-- 文件显示列表 -->
        <table><tr><td>空</td></tr></table>
        <!-- 列表标题模板 -->
        <script id="J_head_tpl" type="text/x-html5-tmpl">
            <tr><th>文件名</th><th>类型</th><th>大小</th><th>预览
</th><th>上传进度</th></tr>
        </script>
        <!-- 列表单行模板 -->
        <script id="J_item_tpl" type="text/x-html5-tmpl">
            <tr>
                <td>{name}</td>
                <td>{type}</td>
                <td>{size}</td>
                <td id="J_{guid}"></td>
                <td>
                          <div class="progressbar">
                                 <div class="progressbar-value"
style="display:none" id="J_p_{guid}"></div>
                          </div>
                     </td>
            </tr>
        </script>
    </section>
</body>
<script>
(function () {
    var multi_file = document.querySelector('input[type=file]'),
    // 获取上传控件
        file_list = document.querySelector('table'),
        // 获取列表控件
        submit = document.querySelector('input[type=submit]'),
         // 获取提交按钮
        form = document.querySelector('form'),
        // 获取页面表单
        slice = Array.prototype.slice,
```

```
            // 获取数组slice原型方法
            form_action = form.getAttribute('action'),
            // 获取表单提交远程地址
            head_tpl = document.getElementById('J_head_tpl').innerHTML,
            // 获取列表标题模板
            item_tpl = document.getElementById('J_item_tpl').innerHTML,
            // 获取列表单行模板
            file_map = {};
            // 缓存文件对象

        var GUID = 0;
        function guid() { return GUID++; };
        // 全球唯一标识符生成函数
        function substitute(template, params) {
        // 工具方法，用对象替换字符串中的占位符
            return ('' + template).replace(/\\?\{([^{}]+)\}/g, function (match, name) {
                return match.charAt(0) === '\\' ? match.slice(1) : (params[name] != null ? params[name] : '');
            });
        };
        function upload_file(file, gid) {
        // 用于上传单个文件
            var xhr = new XMLHttpRequest(),
            // 实例化XMLHttpRequest，用于与后台通信
                formData = new FormData(),
                // 实例化FormData，用于存储表单数据信息
                // 通过传入的gid获取对应上传文件的显示进度条
                  progress = document.getElementById('J_p_' + gid);
            formData.append('file', file);
            // 添加文件数据到formData对象
                // 初始化HTTP请求，设置请求类型、地址、是否异步
            xhr.open('post', form_action, true);
                // 监听XMLHttpRequest对象的load事件，读取完毕后触发
            xhr.addEventListener('load', function () {
                var data = JSON.parse(xhr.responseText);
                // 将返回的字符串转化为对象
                console.log('上传完毕： ' + file.name);
                console.log(data.msg);
            });
                // 监听XMLHttpRequest对象的progress事件，读取完毕后触发
            xhr.upload.addEventListener('progress', function (e) {
                var percent;
                if (e.lengthComputable) {          // 是否可以计算上传字节数
                    percent = (e.loaded / e.total * 100) + '%';
                    // 上传字节数除以总字节数得到上传进度百分比
                    progress.style.width = percent;
                    // 更新上传进度条样式为当前上传比例相同
                    progress.innerHTML = percent;        // 更新上传百分比
                };
            });
```

```
        xhr.send(formData);                                 // 发送数据到服务器
        progress.style.display = '';
    };

    multi_file.addEventListener('change', function (e) {
    // 监听上传控件数据变化
        var files = slice.call(multi_file.files, 0),
        // 将FileList对象转为数组获取forEach方法
            htmls = [head_tpl];                         // 初始化数组，填充列表标题
        file_map = {};                                  // 重置文件缓存对象
        files.forEach(function (file, index) {  // 循环File数组
            var reader,            // reader变量存放FileReader实例
                _guid = guid();                         // 获取全球唯一标识符
            htmls.push(substitute(item_tpl, {
            // 组装 File对象的 name、type、size属性
                name: file.name,
                type: file.type,
                size: file.size,
                guid: _guid
            }));
            if (file.type.toLowerCase().match(/image.*/)) {
            // 用正则表达式判断文件是否为图片类型
                reader = new FileReader();
                // 实例化FileReader对象，用于读取文件数据
                reader.addEventListener('load', function (e) {
                 // 监听FileReader实例的load事件
                    // 获取对应元素，并填充相应图片
                    document.getElementById('J_' + _guid).innerHTML =
'<img title="' + file.name + '" alt="' + file.name + '" src="'
  + e.target.result + '"/>';
                });
                reader.readAsDataURL(file);             // 读取文件为DataURL
            };
            file_map[_guid] = file;        // 存储文件到临时缓存对象，用于提交保存
        });
        file_list.innerHTML = htmls.join('');
        // 填充列表html，显示选择文件
    }, false);
    submit.addEventListener('click', function (e) {
    // 监听提交单击事件
        e.preventDefault();                     // 取消表单提交按钮事件的默认动作
        for (var gid in file_map) {             // 循环文件缓存列表
            upload_file(file_map[gid], gid);
            // 调用upload_file方法，传入文件对象和gid
        };
    }, false);
})();
</script>
</html>
```

利用编辑器打开“012.upload-server”文件夹中的server.js文件，代码如下：

```
var formidable = require("formidable"),
// 引用formidable模块，用于文件保存
    fs = require('fs'),                 // 引用fs模块，用于对文件操作
    http = require("http");             // 引用http模块，用于web服务器
    http.createServer(function (req, res) {            // 创建新服务器
    var form = new formidable.IncomingForm(), // 新建form实例
        files = [];                // 新建files数组，用于存储上传文件实例
    // 所有域名跨域访问均可以通过
    res.setHeader('Access-Control-Allow-Origin', '*');
    // 服务器支持 'OPTIONS, HEAD, GET, POST, PUT, DELETE' 方法
    res.setHeader('Access-Control-Allow-Methods', 'OPTIONS, HEAD, GET, POST, PUT, DELETE');
    // 浏览器Ajax跨域会发送两个请求，第一个请求方法为OPTIONS，预检查服务器
    if (req.method == 'OPTIONS') {
        res.end();
    } else {
        form.on('end', function () {            // 监听form上传结束事件
            res.write(JSON.stringify({          // 写入上传成功状态码和信息
                "code": 200,
                "msg": '上传成功，文件已经保存到服务器，远程路径：' + form.uploadDir
            }));
            res.end();                          // 响应结束
        }).on('file', function (field, file) {  // 监听form文件上传事件
            files.push(file);              // 将上传文件的file实例存入数组
        }).on('error', function (error) {       // 监听form上传出错事件
            res.write(JSON.stringify({          // 写入上传失败状态码和信息
                "code": 500,
                "msg": '上传失败，' + error
            }));
            res.end();
        });
        form.parse(req, function (err) {
        // 调用parse接口解析请求信息并保存文件
            files.forEach(function (item, index) {    // 循环上传的文件数组
                console.log(item.name);         // 在控制台显示文件名
                // 将上传至服务器端的文件名重命名为原文件名
                fs.rename(item.path, form.uploadDir + '\\' + item.name, function (err) { });
            });
        });
    };
}).listen(8080, function () {              // 设置Web服务器监听端口，并启动服务
    // 控制台显示Web服务器启动成功
    console.log('listening on http://localhost:8080');
});
```

5.12.3 代码分析

1. 客户端网页代码分析

在CSS中增加了进度条样式，并调整了上传控件的宽度，见代码第6行~第28行。

HTML代码增加两个模板，分别用于列表的标题和列表的行元素，见代码第43行~第45行和代码第47行~第59行。

代码第66行，通过调用document的querySelector方法获取页面上传控件元素。



querySelector方法根据CSS选择器规范获取文档中的指定元素，更多的应用接口信息可参考网站http://www.w3.org/TR/selectors-api/。

代码第70行和71行，获取存放在script标签内的模板内容。后面会利用函数substitute将对象写入模板，获取对应的HTML结构。

代码第72行定义了一个临时对象，用于存储上传列表的文件对象。

代码第76行~第80行定义函数substitute，传入template和params参数。template为String类型，表示用户的自定义模板。params为Object类型，表示数据对象字面量，使用方法代码如下：

```
var template = "{name} is The New HTML Standard";
var params = { "name": "HTML5" };
console.log(substitute(template, params));  // "HTML5 is The New
HTML Standard"
```

代码第81行~第106行定义本示例关键函数upload_file，用于实现无刷新上传单个数据文件。

代码第82行实例化XMLHttpRequest对象，用于对HTTP协议进行访问，是Ajax技术的核心。



Ajax表示异步JavaScript和XML，是一种利用XMLHttpRequest与服务器进行通信，通过JavaScript无刷新改变页面的Web技术。在各种浏览器上实现时具有一定的差异，IE下使用ActiveXObject。

代码第83行实例化formData对象，用于存储表单数据。HTML 5的formData的出现，方便了XMLHttpRequest的请求头设置，在每次传递过程中会配置适当的HTTP头信息。

代码第86行，调用formData对象的append方法，将传入的文件对象推入刚刚实例化的formData对象中。

代码第88行，调用XMLHttpRequest的open方法，传入请求方法类型、上传服务器地址和是否异步三个参数值。

代码第90行~第94行监听XMLHttpRequest的load状态，在每次上传完毕以后触发。代码第91行获取返回的responseText，并通过JSON的parse方法将返回字符串转换为数据对象。

代码第96行~第103行监听XMLHttpRequest的progress状态。代码第98行通过监听，回调函数返回数据e，该值为ProgressEvent对象。

ProgressEvent接口继承至Event，同时增加了三个属性，接口代码如下。

```
interface ProgressEvent : Event {
  readonly attribute boolean lengthComputable;
  readonly attribute unsigned long long loaded;
  readonly attribute unsigned long long total;
```

```
};
```

- lengthComputable：上传字节数可计算时返回true。
- loaded：已上传字节数。
- total：总字节数。

代码第104行，调用XMLHttpRequest的send方法，发送formData数据到远程服务器。

2．服务器端代码分析

服务器端代码也做了部分调整，增加了HTTP响应头信息和HTTP请求方法OPTIONS的处理逻辑。

XMLHttpRequest跨源资源共享需要服务器端在响应时发送“Access-Control-Allow-Origin”头信息，示例代码中将该信息的值设置为“*”，表示允许所有域请求。同时设置“Access-Control-Allow-Methods”头信息为“OPTIONS,HEAD,GET,POST,PUT,DELETE”，表示服务器允许接收字符串中出现的HTTP方法，代码如下所示：

```
res.setHeader('Access-Control-Allow-Origin', '*');
res.setHeader('Access-Control-Allow-Methods', 'OPTIONS, HEAD, GET, POST,
 PUT, DELETE');
```

代码第12行~第13行判断HTTP请求是否是OPTIONS类型，如果是则立刻返回响应。XMLHttpRequest在做跨源资源请求时，会预先向远端服务器发送一个OPTIONS方法的请求，用于预检测服务器是否可以安全发送。



更多关于OPTIONS方法在XMLHttpRequest的信息可以参考http://www.w3.org/TR/cors/和http://hacks.mozilla.org/2009/07/cross-site-xmlhttprequest-with-cors/中的两篇文章。

5.12.4 相关知识

HTTP规范中定义了8种请求方法，分别如下。

- GET：向指定资源发出请求。
- POST：向指定资源提交数据处理请求。
- HEAD：向指定资源发送请求，只返回响应消息头中的元信息，不返回响应内容。
- PUT：向指定资源位置上传其最新内容。
- DELETE：请求服务器删除Request-URI所标识的资源。
- OPTIONS：返回资源服务器支持的HTTP请求方法。
- TRACE：回显服务器收到的请求，主要用于测试或诊断。
- CONNECT：协议中预留给能够将连接改为管道方式的代理服务器。



5.13 示例13 拖曳上传文件

5.13.1 示例效果

接着之前的上传示例，本例进一步通过HTML 5增强用户体验，这次新加入的功能是拖曳。用户可以直接将文件夹中的文件拖入列表框，鼠标松开后，页面列表呈现出刚刚拖入文件的信息，与通过上传控件选择文件呈现的效果相同。

启动上传Web服务器，本例继续沿用示例12的上传Web服务器。使用Chrome浏览器打开网页文件，运行效果如图5.44所示。



图5.44 使用Chrome打开网页文件

选中文件夹中的文件，拖入浏览器列表框。移入时，列表边框变为黄色虚线，如图5.45所示。



图5.45 选中文件拖入浏览器

松开鼠标，列表显示文件的“文件名”、“类型"、“大小”、“预览”、“上传进度”信息，运行效果如图5.46所示。

选择文件 未选择文件 提交

文件名	类型	大小	预览	上传进度
Lighthouse.jpg	image/jpeg	561276		
Penguins.jpg	image/jpeg	777835		
Tulips.jpg	image/jpeg	620888		
工作.xlsx	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	9879		
内容.txt	text/plain	0		

图5.46 显示的文件信息

单击“提交”按钮，文件被依次上传到服务器，如图5.47所示。

选择文件 未选择文件 提交

文件名	类型	大小	预览	上传进度
Lighthouse.jpg	image/jpeg	561276		100%
Penguins.jpg	image/jpeg	777835		100%
Tulips.jpg	image/jpeg	620888		100%
工作.xlsx	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	9879		100%
内容.txt	text/plain	0		100%

图5.47 单击“提交”按钮上传文件

5.13.2 代码设计

利用编辑器打开“013.拖曳上传文件.html”文件，代码如下：

```
<!doctype html>
<html>
<head>
```

```
    <style>
        /* 同示例012 */
        .hover                                      /* 文件拖入列表提示样式 */
        {
            border:4px dashed orange;
            height:190px;
            margin-top:10px;
        }
    </style>
</head>
<body>
    <header><h2>拖曳上传文件</h2></header>
    /* 同示例012 */
</body>
<script>
(function () {
    /* 同示例012 */
    function render_list(files) {                   // 初始化数组，填充列表标题
        file_map = {};                              // 重置文件缓存对象
        files.forEach(function (file, index) {     // 循环File数组
            var reader,                          // reader变量存放FileReader实例
                _guid = guid();                  // 获取全球唯一标识符
            htmls.push(substitute(item_tpl, { // 组装 File属性为html
                name: file.name,
                type: file.type,
                size: file.size,
                guid: _guid
            }));
            if (file.type.toLowerCase().match(/image.*/)) {
            // 用正则表达式判断文件是否为图片类型
                reader = new FileReader();
                // 实例化FileReader对象，用于读取文件数据
                reader.addEventListener('load', function (e) {
                // 监听FileReader实例的load事件
                    // 获取对应元素，并填充相应图片
                    document.getElementById('J_' + _guid).innerHTML =
'<img title="' + file.name + '" alt="'       + file.name + '" src="'
 + e.target.result + '"/>';
                });
                reader.readAsDataURL(file);
                // 读取文件为DataURL
            };
            file_map[_guid] = file;
            // 存储文件到临时缓存对象，用于提交保存
        });
        file_list.innerHTML = htmls.join('');
    };
    multi_file.addEventListener('change', function (e) {
    // 监听上传控件数据变化
        render_list(slice.call(multi_file.files, 0));
        // 将FileList对象转为数组获取forEach方法
```

```
47      }, false);
48      submit.addEventListener('click', function (e) {
        // 监听提交单击事件
49          e.preventDefault();
            // 取消表单提交按钮事件的默认动作
50          for (var gid in file_map) {
      // 循环文件缓存列表调用upload_file上传文件
51              upload_file(file_map[gid], gid);
52          };
53      }, false);
54      file_list.addEventListener('dragenter', function (e) {
        // 按下鼠标，该元素被拖入时触发
55          e.preventDefault();
56          this.className = 'hover';
57      }, false);
58      file_list.addEventListener('dragleave', function (e) {
        // 当鼠标拖曳离开该元素时触发
59          e.preventDefault();
60          this.className = '';
61      }, false);
62      file_list.addEventListener('dragover', function (e) {
        // 当鼠标拖曳该元素移动时触发
63          e.preventDefault();
64      }, false);
65      file_list.addEventListener('drop', function (e) {
        // 鼠标拖曳该元素释放时触发
66          e.preventDefault();
67          this.className = '';
68          render_list(slice.call(e.dataTransfer.files, 0));
69      }, false);
70  })();
71  </script>
72  </html>
```



上传服务器端的逻辑代码沿用示例12。

5.13.3 代码分析

代码第21行~第44行定义函数render_list，用于读取文件构建时显示列表文档。

代码第54~69行分别监听了dragenter、dragleave、dragover、drop事件。

当文件被拖入列表时，触发dragenter事件，此时设置列表的样式为hover，显示层上表现为出现黄色虚线框。

当鼠标拖曳并且未松开，离开列表时，触发dragleave事件，还原列表样式并且黄色虚线框消失。

当鼠标拖曳文件在列表上时，触发dragover事件（不管鼠标是否移动都会多次触发），此时必须调用传入回调函数事件对象的preventDefault方法，阻止默认行为，否则无法触发

drop事件。

当松开鼠标时，触发drop事件，回调函数获取数据对象e，该对象为DragEvent（继承至MouseEvent），DragEvent接口代码如下：

```
interface DragEvent : MouseEvent {
  readonly attribute DataTransfer? dataTransfer;
};
```

通过DataTransfer的files属性获取拖曳的文件列表，传入render_list函数进行展现。DataTransfer接口代码如下：

```
interface DataTransfer
{
  attribute DOMString dropEffect;          // 设置用户完成放置后的动作
  attribute DOMString effectAllowed;       // 设置允许的拖动操作
  readonly attribute DataTransferItemList items;
  // 存取DataTransferItem对象接口
  void setDragImage(Element image, long x, long y);
  // 设置拖曳元素时随鼠标移动的图片
  readonly attribute DOMString[] types;
  // 返回dragstart触发时为元素的存储数据类型
  DOMString getData(DOMString format);
  // 返回指定数据，如果数据不存在，则返回空字符串
  void setData(DOMString format, DOMString data);//
  为元素添加数据，dragstart事件触发时为拖曳元素存储数据
  void clearData(optional DOMString format);
  // 删除指定格式的数据，如果不指定格式则删除全部数据
  readonly attribute FileList files;
  // 如果拖曳的是系统文件，返回正在拖曳的文件列表
};
```



第 6 章 Canvas图画大演练

Canvas是一种全新的HTML元素，可以通过JavaScript语言来绘制图形，例如画图、处理图像合成、制作动画，甚至是游戏。Canvas被认为是HTML 5最为期待的特性之一。本章将从常规图形绘制、文字绘制、颜色渐变和图片生成出发，介绍Canvas的各种技术，接着会通过制作动画计时器和相片的360°旋转功能了解更多的Canvas使用技巧，最后给读者带来使用Canvas制作数据报表和动画的实际案例。

本章知识点：

- 绘制图形
- 制作动画
- 制作HTML 5版销售数据表



6.1 示例1 绘制图形（矩形和圆形）

6.1.1 示例效果

画图软件已经不再是本地应用的专利了，越来越多的网站提供了在线版的画图应用。有的通过Flash实现，有的通过HTML 5实现。本例给出了用HTML 5实现的简单画板。画板分成两个部分，第一部分为矩形，另外一个为圆形。两部分都有一个调节区和一个画布区。画布区默认绘制一个对应图形，调整颜色控件或拖动滑块同步作用于绘制的图形。通过本示例可以了解Canvas图形绘制接口的使用。

使用Chrome浏览器打开网页文件，运行效果如图6.1所示。

单击“矩形”颜色控件，设置画布中黄色矩形颜色为红色，然后单击“圆形”颜色控件，设置画布中蓝色圆形颜色为绿色，效果如图6.2所示。

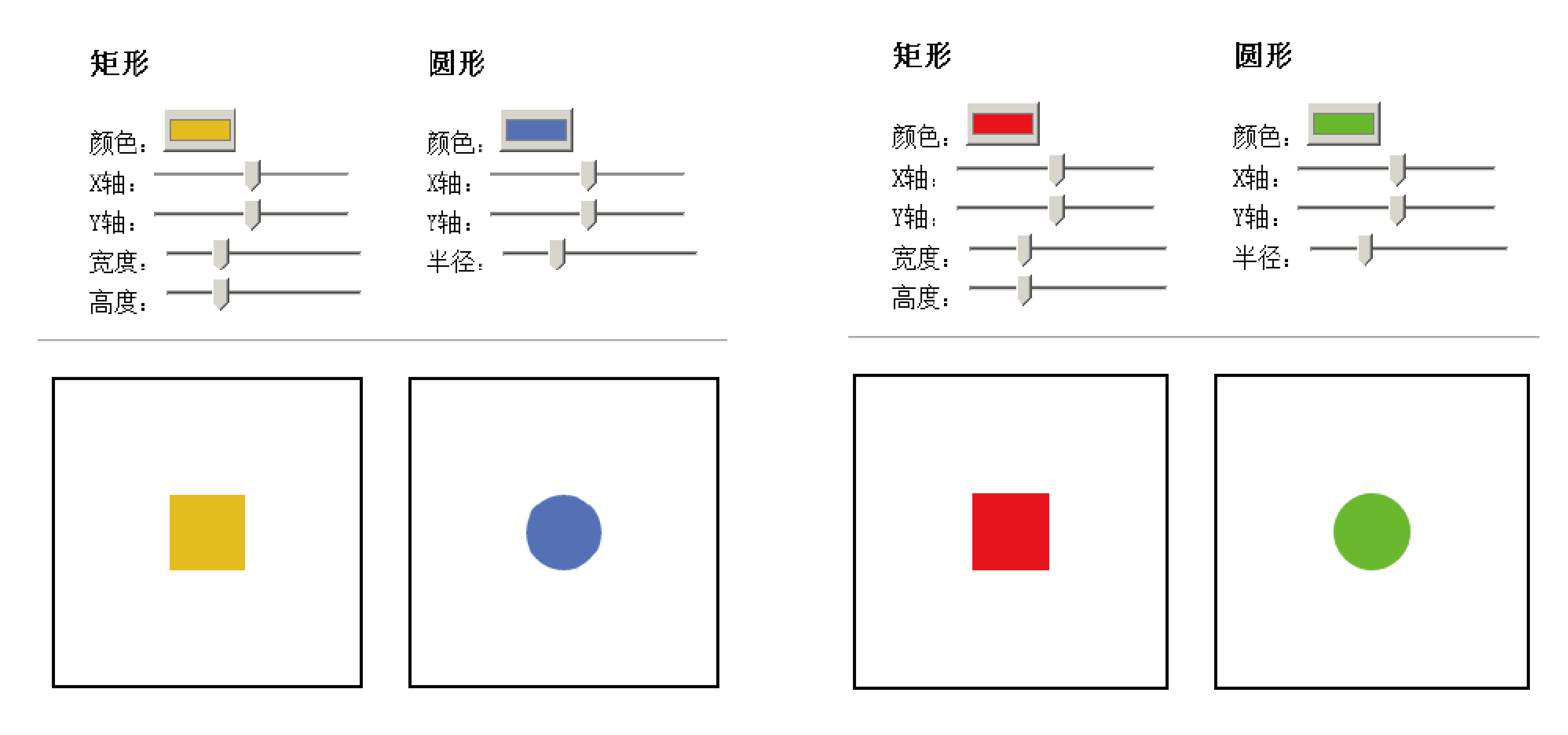


图6.1　使用Chrome打开网页文件　　图6.2　调整“矩形”和“圆形”颜色控件

6.1.2　代码设计

利用编辑器打开“001.绘制图形：矩形和圆形.html”文件，代码如下：

```
<!DOCTYPE HTML>
<html>
<head>
    <style>
        ul{ float: left; list-style: none; }
        hr{ clear: both; }
        canvas{ border:2px solid black; float:left; margin:15px; }
    </style>
</head>
<body>
    <header><h2>绘制图形：矩形和圆形</h2></header>
    <!-- 矩形 Canvas画布 设置区 -->
    <ul>
        <li><h3>矩形</h3></li>
        <li>颜色：<input id="color_rect" type="color" value="#eabb02"
 /></li>
        <li>X轴：<input id="x_rect" type="range" min="0" max="150"
value="75" step="1" /></li>
        <li>Y轴：<input id="y_rect" type="range" min="0" max="150"
value="75" step="1" /></li>
        <li>宽度：<input id="width_rect" type="range" min="0" max="200"
value="50" step="1" /></li>
        <li>高度：<input id="height_rect" type="range" min="0" max="200"
value="50" step="1" /></li>
    </ul>
    <!-- 圆形 Canvas画布 设置区 -->
    <ul>
        <li><h3>圆形<h3></li>
        <li>颜色：<input id="color_circle" type="color"
value="#506cc0"/></li>
        <li>X轴：<input id="x_circle" type="range" min="0" max="200"
```

```
value="100" step="1" /></li>
        <li>Y轴：<input id="y_circle" type="range" min="0" max="200"
value="100" step="1" /></li>
        <li>半径：<input id="radius_circle" type="range" min="0"
max="100" value="25" step="1" /></li>
    </ul>
    <hr />
    <!-- 矩形 Canvas画布 -->
    <canvas id="canvas_rect" width="200" height="200"></canvas>
    <!-- 圆形 Canvas画布 -->
    <canvas id="canvas_circle" width="200" height="200"></canvas>
</body>
<script>
    (function () {
        // 获取矩形Canvas画布绘图上下文
        var content_rect = document.getElementById('canvas_rect').
getContext("2d"),
        // 获取圆形Canvas画布绘图上下文
            canvas_circle = document.getElementById('canvas_circle').
getContext("2d");
        function draw_rect() {                      // 获取控件数据绘制矩形
            content_rect.clearRect(0, 0, 300, 300);
            // 清空给定矩形
            content_rect.fillStyle = color_rect.value;
            // 填充矩形画布原色
            content_rect.fillRect(                  // 填充矩形
                parseInt(x_rect.value),             // 矩形左上角的x坐标
                parseInt(y_rect.value),             // 矩形左上角的y坐标
                parseInt(width_rect.value),         // 矩形的宽度，以像素计
                parseInt(height_rect.value)         // 矩形的高度，以像素计
            );
        };
        var color_rect = document.getElementById('color_rect'),
        // 获取矩形颜色选择元素
            x_rect = document.getElementById('x_rect'),
            // 获取矩形x轴滑块元素
            y_rect = document.getElementById('y_rect'),
            // 获取矩形y轴滑块元素
            width_rect = document.getElementById('width_rect'),
            // 获取矩形宽度滑块元素
            height_rect = document.getElementById('height_rect');
            // 获取矩形高度滑块元素
        // 循环矩形设置元素，绑定数据变更change事件
        [color_rect, x_rect, y_rect, width_rect, height_rect].
forEach(function (item) {
            item.addEventListener('change', draw_rect, false);
        });
        draw_rect();                                // 绘制默认矩形
        function draw_circle() {                    // 获取控件数据绘制圆形
            canvas_circle.clearRect(0, 0, 300, 300);
            // 清空给定矩形
            canvas_circle.fillStyle = color_circle.value;
            // 填充矩形画布原色
            canvas_circle.beginPath();              // 起始一条路径
            canvas_circle.arc(                      // 创建圆形
```

```
66                    parseInt(x_circle.value),     // 圆中心的 x 坐标
67                    parseInt(y_circle.value),     // 圆中心的 y 坐标
68                    parseInt(radius_circle.value),       // 圆的半径
69                    0,                            // 起始角，以弧度计
70                    2 * Math.PI,                  // 结束角，以弧度计
71                    true                          // 逆时针或顺时针绘图
72                                           //（false：顺时针，true：逆时针）
73                );
74                canvas_circle.closePath(); // 创建从当前点回到起始点的路径
75                canvas_circle.fill();             // 填充当前绘图
76            };
77            var color_circle = document.getElementById('color_circle'),
              // 获取圆形颜色选择元素
78                x_circle = document.getElementById('x_circle'),
                  // 获取圆形x轴滑块元素
79                y_circle = document.getElementById('y_circle'),
                  // 获取圆形y轴滑块元素
80                radius_circle = document.getElementById('radius_circle');
                  // 获取圆形半径滑块元素
81                       // 循环圆形设置元素，绑定数据变更change事件
82            [color_circle, x_circle, y_circle, radius_circle].
   forEach(function (item) {
83                item.addEventListener('change', draw_circle, false);
84            });
85            draw_circle();                        // 绘制默认圆形
86        })();
87    </script>
88    </html>
```

6.1.3 代码分析

代码第12~20行是矩形调节区的HTML结构，第21~28行是圆形调整区的HTML结构。

代码第31行和第33行分别是矩形和圆形的画布。

代码第38行和第40行，分别从文档中获取对应画布的上下文。

代码第41行~第50行为函数draw_rect，用于在画布上绘制矩形。该函数第42行代码清空画布上固定矩形区域的内容，就像平时所用的橡皮擦。clearRect方法拥有4个参数，语法如下：

```
context.clearRect(x,y,width,height)
```

- x：要清除的矩形左上角的x轴坐标。
- y：要清除的矩形左上角的y轴坐标。
- width：要清除矩形的宽度，以像素计。
- height：要清除矩形的高度，以像素计。

代码第43行，获取矩形调节区颜色控制的值，设置矩形画布的颜色。

代码第44行~第49行填充一个矩形，fillRect方法拥有4个参数，参数用法与clearRect方法相同，语法如下：

```
context.fillRect(x,y,width,height)
```

代码第51~55行，获取矩形调节区颜色控件和各滑块元素。

代码第57~59行，监听矩形调节区各控件的change事件。当各控件的值发生变化时，调用函数draw_rect重新绘制矩形。

代码第60行，调用函数draw_rect，使用矩形调节区各控件的默认值绘制默认矩形。

圆形区的代码基本与矩形区的相似，只是在调用画布上下文的方法上略有不同，绘制圆形通过draw_circle函数实现。

代码第64行，调用方法beginPath，告诉画布开始一条新的路径。假使不调用该方法，每次绘制的图形也会被重叠。

代码第65~73行，调用方法arc以给定的坐标点为中心点以指定半径画一条弧，语法如下：

```
context. arc(x,y,radius,startAngle,endAngle,counterclockwise)
```

- x，y：弧圆心坐标。
- radius：弧半径。
- startAngle，endAngle：弧的开始点和结束点。
- counterclockwise：设置弧沿逆时针或顺时针绘制。

代码第74行调用closePath方法关闭之前打开的路径。第75行调用fill方法，使用设置的fillStyle属性颜色填充绘制的路径。

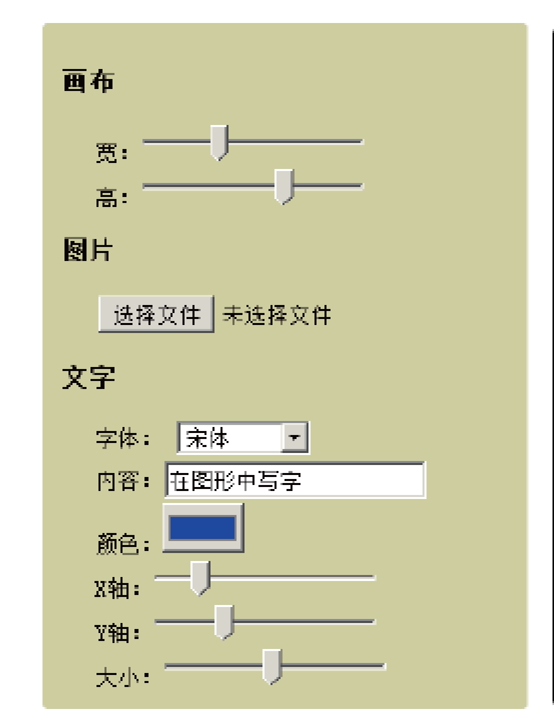


6.2 示例2 在图形中写字

6.2.1 示例效果

随着互联网的发展，图片已成为人们分享信息的一种重要手段，但随之而来的图片盗用问题成为了阻碍用户分享的一大障碍。目前，被认为行之有效的方法是给用户上传的图片加上水印，防止图片被盗用。本例提供了HTML 5加水印的解决方案。

使用Chrome浏览器打开网页文件，运行效果如图6.3所示。



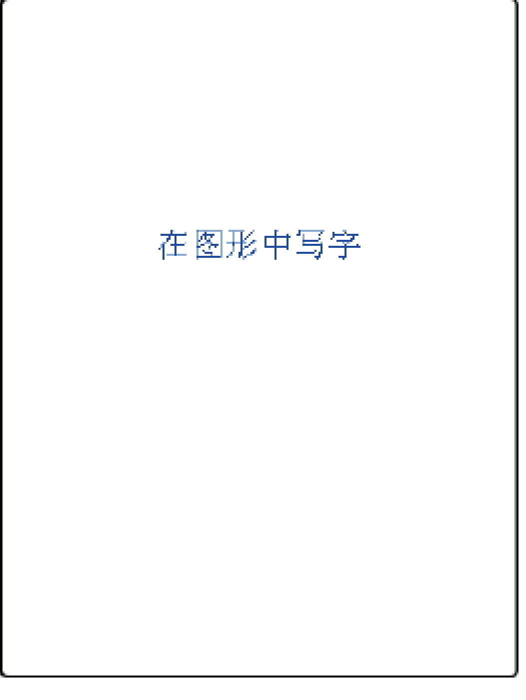


图6.3 使用Chrome打开网页文件

单击“选择文件”按钮，选择任意图片并打开（示例中选择了“考拉”图），效果如图6.4所示。

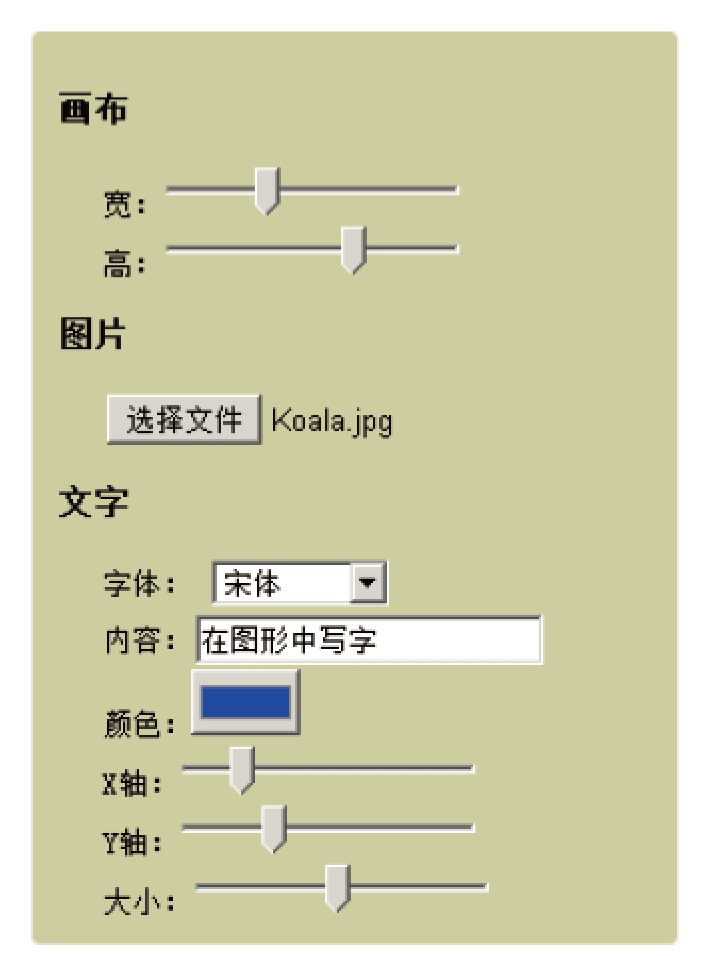




图6.4 单击“选择文件”按钮选择一张图片并打开

在画布设置区调节“宽高”滑块为最大值，然后将光标移入画布框，此时光标样式变为拖动状态。拖动画布框中的图片，直到合适位置后松开，效果如图6.5所示。

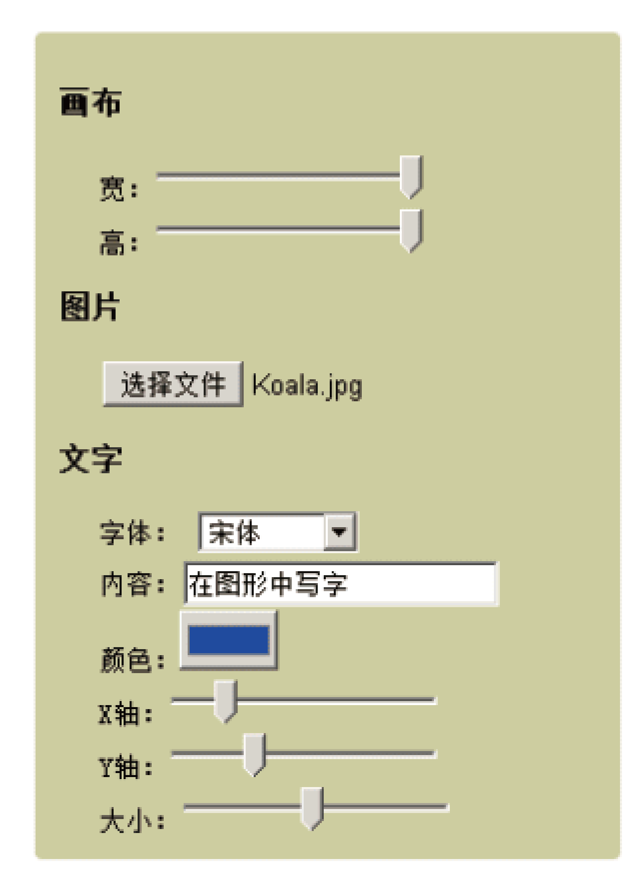




图6.5 调整画布大小并拖动图片到合适位置

此时画布内出现一个完整的考拉头像。接下来给图片打上水印“我是考拉”，并将水印移动到图片的右下角。调整文字的字体、颜色和大小，效果如图6.6所示。

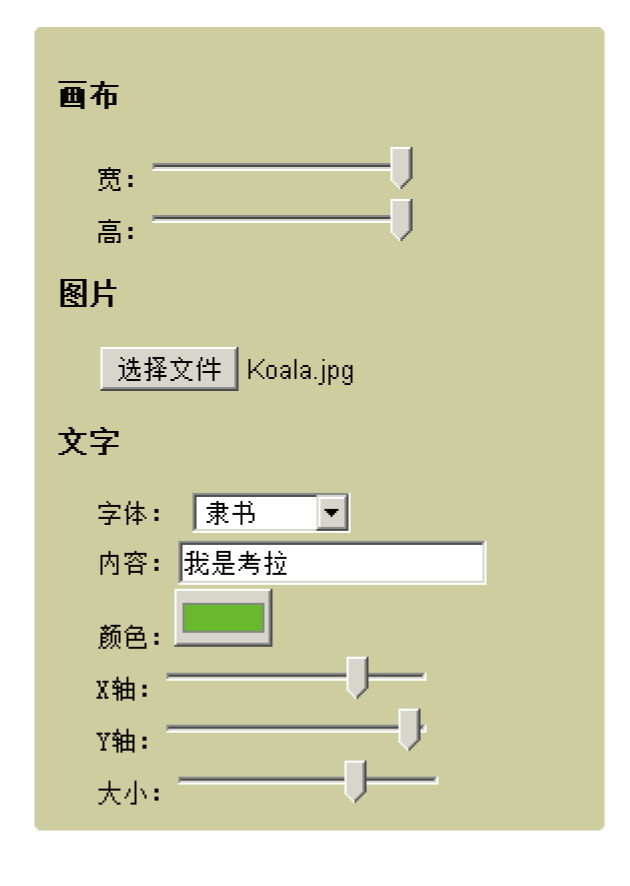




图6.6 调整文字内容和样式

6.2.2 代码设计

利用编辑器打开“002.在图形中写字.html”文件，代码如下：

```
<!DOCTYPE HTML>
<html>
<head>
    <style>
        ul                                              /* 画布调节区样式 */
        {
            float: left;
            list-style: none;
            font-size:13px;
            border: 1px solid #cccc99;
            border-radius: 3px;
            -moz-border-radius: 3px;
            -webkit-border-radius: 3px;
            background-color: #cccc99;
            padding:10px;
        }
        hr{ clear: both; }
        canvas                                          /* 画布样式 */
        {
            border:2px solid black;
            float:left;
            margin:15px; font-family:
            border-radius: 5px;
            -moz-border-radius: 5px;
            -webkit-border-radius: 5px;
        }
```

```
        .item                                  /* 调节区元素行样式 */
        {
            padding-left:20px;
        }
    </style>
</head>
<body>
    <header><h2>在图形中写字</h2></header>
    <section>
        <!-- 矩形 Canvas画布 设置区 -->
        <ul>
            <li><h3>画布</h3></li>
            <!-- 画布长宽调节区 -->
            <li class="item">宽: <input id="width_canvas" type="range"
 min="200" max="500" value="300"
step="1" /></li>
            <li class="item">高: <input id="height_canvas" type="range"
min="200" max="500" value="395" step="1" /></li>
            <li><h3>图片</h3></li>
            <!-- 图片选择区 -->
            <li class="item"><input id="file_img" type="file" value="
在图形中写字" /></li>
            <!-- 文字设置区 -->
            <li><h3>文字</h3></li>
            <li class="item">字体:
                <select id="family_font">
                    <option value="宋体">宋体</option>
                    <option value="黑体">黑体</option>
                    <option value="幼圆">幼圆</option>
                    <option value="微软雅黑">微软雅黑</option>
                    <option value="楷体">楷体</option>
                    <option value="隶书">隶书</option>
                    <option value="方正姚体">方正姚体</option>
                    <option value="方正舒体">方正舒体</option>
                    <option value="华文彩云">华文彩云</option>
                </select>
            </li>
            <li class="item">内容: <input id="text_font" value="在图形中
写字" maxlength=16 /></li>
            <li class="item">颜色: <input id="color_font" type="color"
value="#0000ff" /></li>
            <li class="item">X轴: <input id="x_font" type="range"
min="0" max="500" value="90" step="1"
/></li>
            <li class="item">Y轴: <input id="y_font" type="range"
min="0" max="500" value="150" step="1" /></li>
            <li class="item">大小: <input id="size_font" type="range"
min="1" max="40" value="20" step="1" /></li>
        </ul>
        <!-- 矩形 Canvas画布 -->
        <canvas id="canvas"></canvas>
```

```
    </section>
</body>
<script>
    (function () {
        var canvas = document.getElementById('canvas'),
        // 获取Canvas画布元素
            context = canvas.getContext("2d"),
            // 获取Canvas元素上下文
            width_canvas = document.getElementById('width_canvas'),
            // 画布宽
            height_canvas = document.getElementById('height_canvas'),
            // 画布长
            file_img = document.getElementById('file_img'),
            // 图片选择
            text_font = document.getElementById('text_font'),
             // 文字内容
            color_font = document.getElementById('color_font'),
            // 文字颜色
            x_font = document.getElementById('x_font'),
            // 文字x轴坐标
            y_font = document.getElementById('y_font'),
            // 文字y轴坐标
            size_font = document.getElementById('size_font'),
            // 文字大小
            family_font = document.getElementById('family_font'),
            // 文字字体
            img = new Image();
             // 新建图片元素实例
        function draw(e, x, y) {
        // 绘制图片和文字
            // 清空画布指定矩形区域内容
            context.clearRect(0, 0, parseInt(canvas.width),
parseInt(canvas.height));
            canvas.width = parseInt(width_canvas.value);
            // 设置画布宽
            canvas.height = parseInt(height_canvas.value);
            // 设置画布高
            context.drawImage(img, x || move_x, y || move_y);
            // 填充图片到画布
            context.fillStyle = color_font.value;
             // 设置文字颜色
            context.textAlign = 'left';
             // 设置文字水平对齐方式
            context.font = size_font.value + "px " + family_font.value;
            // 设置文字大小和字体
            // 填充文字到画布指定区域
            context.fillText(text_font.value, parseInt(x_font.value),
 parseInt(y_font.value));
        };
        //绑定文字内容文本框keyup事件，当键盘按键释放时触发
        text_font.addEventListener('keyup', draw, false);
```

```
        // 绑定数值区域选择控件change事件，当数值变化时触发draw函数
        [color_font, x_font, y_font, size_font, width_canvas, height_canvas, family_font].forEach(function (item) {
            item.addEventListener('change', draw, false);
        });
        // 绑定图片load事件，当图片加载完毕后触发
        img.addEventListener('load', draw, false);
        //绑定长传控件change事件，当路径发生变化时触发
        file_img.addEventListener('change', function () {
            var files = this.files,
             // 获取文件列表
                reader;
            for (var i = 0, length = files.length; i < length; i++) {
                if (files[i].type.toLowerCase().match(/image.*/)) {
                 // 用正则表达式判断文件类型是否为图片类型
                    reader = new FileReader();
                     // 实例化FileReader对象
                    reader.addEventListener('load', function (e) {
                     // 监听FileReader实例的load事件
                        img.src = e.target.result;
                        // 设置图片内容
                    });
                    reader.readAsDataURL(files[i]);
                    // 读取图片文件为dataURL格式
                    canvas.style.cursor = 'move';
                    // 设置光标为移动样式
                    break;
                };
            };
        }, false);
        var move_x = 0, move_y = 0;
        // 临时存储图片x、y轴偏移量
        function canvas_mousemove(e) {
        // 当鼠标拖动图片时触发
            // 计算图片拖动后的x轴位置
            move_x = e.clientX - canvas.$mousedown_x + canvas.$mouseup_move_x;
            // 计算图片拖动后的y轴位置
            move_y = e.clientY - canvas.$mousedown_y + canvas.$mouseup_move_y;
            // 按照计算后的坐标位置重新绘制图片和文字
            draw(null, move_x, move_y);
        };
        canvas.addEventListener('mousedown', function (e) {
        // 当单击画布区时触发
            if (img.src.length) {
             // 判断画布区内是否已经存在图片
                canvas.$mousedown_x = e.clientX;
                // 缓存当前鼠标x轴坐标
                canvas.$mousedown_y = e.clientY;
```

```
                        // 缓存当前鼠标y轴坐标
139                     // 监听画布区鼠标拖动事件
140                     canvas.addEventListener('mousemove', canvas_mousemove,
 false);
141                 };
142             }, false);
143             document.addEventListener('mouseup', function (e) {
// 当鼠标在文档内释放后触发
144                 canvas.$mouseup_move_x = move_x;
                    // 缓存拖动后图片x轴坐标
145                 canvas.$mouseup_move_y = move_y;
                    // 缓存拖动后图片y轴坐标
146                 // 移除对画布鼠标拖动监听事件
147                 canvas.removeEventListener('mousemove', canvas_mousemove,
false);
148             }, false);
149             // 阻止文档内容选择事件，避免拖动时触发内容选择造成不便
150             document.addEventListener('selectstart', function (e)
{ e.preventDefault() }, false);
151             draw();
                // 绘制默认内容
152         })();
153 </script>
154 </html>
```

6.2.3 代码分析

本例涉及多个逻辑功能点，将代码功能分成三大块进行分析。

1. 在画布中显示本地图片

代码第88行创建一个图片实例赋予变量img。

代码第109行监听变量img的load事件，当图片的src属性发生改变时触发函数draw。

代码第111行~第125行添加对上传控件change事件的监听，当上传的控件值改变时触发事件。

代码第112行定义变量files存储上传文件列表。

代码第114行~第124行循环文件列表，判断文件类型是否为图片类型。遇到图片文件，通过FileReader读取文件内容。当文件读取完毕，重新设置变量img的src属性时，待图片加载完毕以后触发函数draw。函数draw中绘制图片方法代码如下：

```
context.drawImage(img, x || move_x, y || move_y);
```

drawImage语法如下：

```
drawImage(image, sourceX, sourceY, sourceWidth, sourceHeight,destX, destY,
 destWidth, destHeight)
```

- image：所要绘制的Image元素。
- sourceX，sourceY：图像在画布左上角坐标位置。

- sourceWidth，sourceHeight：图像的宽和高。
- destX，destY：所要绘制的图像区域的左上角的画布坐标。
- destWidth，destHeight：图像区域所要绘制的画布大小。



在画布中绘制图像必须在图像加载完毕以后执行，即在图像的load事件回调中进行，否则绘制为空图。

2. 在画布中写入文字

函数draw有两个功能，一个是绘制图片，另外一个是在画布上写入文字。
代码第91行，清空对应画布大小区域的内容。
代码第92行、第93行，读取画布调节区宽高滑块的值，重新设置画布宽高。
代码第95行~第97行，分别设置画布文字的颜色、水平位置、大小和字体。
代码第99行，调用画布的fillText方法写入文字，fillText语法如下：

```
context.fillText(text, x, y, maxWidth);
```

- text：文本内容。
- x：相对于画布的x轴坐标。
- y：相对于画布的y轴坐标。
- maxWidth：可选，允许的最大文本宽度，以像素计。

3. 光标拖动画布中的图片

使用脚本完成拖动元素效果，首先要明白元素的三种事件。

- mousedown：事件会在鼠标按键被按下时发生。
- mouseup：事件会在鼠标按键被松开时发生。
- mousemove：事件会在鼠标指针移动时发生。

本例实现画布中图片拖动的效果见代码第126~148行，下面分析相关的代码逻辑。
代码第126行，定义了两个初始化为0的变量，用于记录图片拖动后的坐标偏移量。

代码第136行~第141行，单击画布时被触发。mousedown被触发后，先判断画布中是否已经存在图片，假使已经上传图片（即img变量的src属性不为空），获取鼠标指针位置相对于当前窗口的x轴坐标和y轴坐标，并将两个坐标值缓存在canvas元素变量上。最后监听canvas元素的mousemove事件，绑定canvas_mousemove函数。

代码第144行~第147行，鼠标在文档内松开时被触发。mouseup被触发后，将move_x，move_y两个图片坐标偏移量缓存在canvas元素变量上，并将移除mousedown时绑定在canvas元素上的mousemove事件进行监听。

代码第127行~第134行是完成拖动效果的关键。拖动时计算图画和画布的相对位置，代码如下：

```
move_x = e.clientX - canvas.$mousedown_x + canvas.$mouseup_move_x;
move_y = e.clientY - canvas.$mousedown_y + canvas.$mouseup_move_y;
```

首先分析如何得到move_x，即当前图片拖动后相对于画布x轴坐标（move_y的计算方法与move_x类似）。

- e.clientX：拖动时鼠标指针位置相对于当前窗口的x轴坐标。
- canvas.$mousedown_x：拖动开始时鼠标指针位置相对于当前窗口的x轴坐标。
- canvas.$mouseup_move_x：上次拖动结束后图片相对于画布的偏移量，默认未拖动，为0。



代码分析并未包括所有源码，未提及的可以参考代码后方的注释。

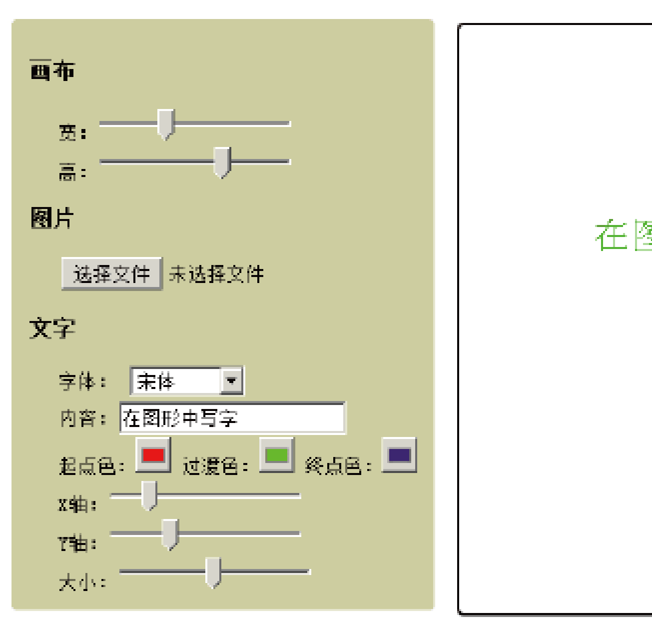


6.3 示例3 在画布中使用渐变色

6.3.1 示例效果

是否觉得示例2的水印文字变化过于单调，本例将在其基础上，运用Canvas增加文字渐变处理。示例中将原来的“颜色”一栏变化为“起始色”、“过渡色”和“终止色”，并采用线性渐变以直线为渐变轴处理文字。

使用Chrome浏览器打开网页文件，运行效果如图6.7所示。



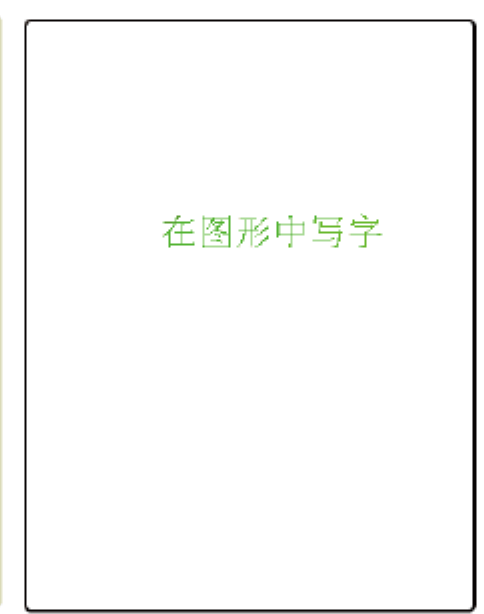


图6.7 使用Chrome打开网页文件

选择“图片库”中的“郁金香.jpg”并打开。调整画布宽度、文字内容、文字XY轴和文字大小，效果如图6.8所示。

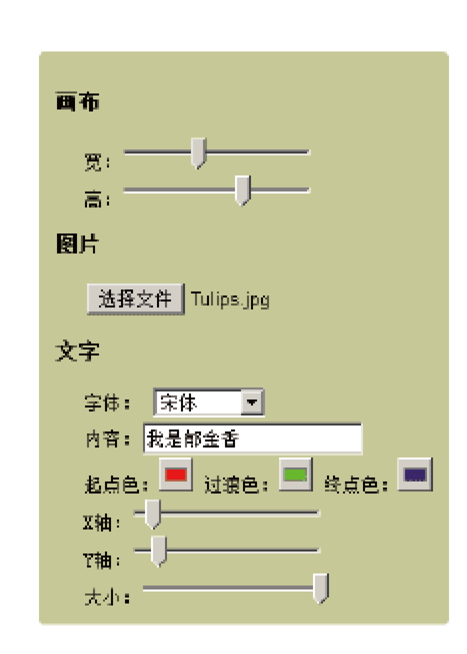




图6.8 选择图片并调整相关设置

6.3.2 代码分析

本例代码基本与示例2相同，下面就不同部分做分析。

去掉了原有的文字“颜色”控件后，新增了三个颜色选择控件，代码如下：

```
起点色：<input id="color_font_begin" type="color" value="#ff0000" />
过渡色：<input id="color_font_middle" type="color" value="#00ff00" />
终点色：<input id="color_font_end" type="color" value="#400080" />
```

draw函数增加了文字渐变色处理，代码如下：

```
01 function draw(e, x, y) {
   // 绘制图片和文字
02     // 清空画布指定矩形区域内容
03     context.clearRect(0, 0, parseInt(canvas.width), parseInt(canvas.
   height));
04     canvas.width = parseInt(width_canvas.value);
       // 设置画布宽
05     canvas.height = parseInt(height_canvas.value);
       // 设置画布高
06     context.drawImage(img, x || move_x, y || move_y);
       // 填充图片到画布
07     // 创建线性渐变
08     var gradient = context.createLinearGradient(0, 0, canvas.width,
   canvas.height);
09     gradient.addColorStop("0.0", color_font_begin.value);
       // 在渐变起点添加一个颜色变化
10     gradient.addColorStop("0.5", color_font_middle.value);
       // 在渐变中间点添加一个颜色变化
11     gradient.addColorStop("1.0", color_font_end.value);
       // 在渐变终点添加一个颜色变化
12     context.fillStyle = gradient;
       // 用渐变填色
13     context.textAlign = 'left';
       // 设置文字水平对齐方式
14     context.font = size_font.value + "px " + family_font.value;
       // 设置文字大小和字体
15     // 填充文字到画布指定区域
16     context.fillText(text_font.value, parseInt(x_font.value),
    parseInt(y_font.value));
17 };
```

draw函数第8行代码调用context的createLinearGradient方法，该方法创建一个线性颜色渐变对象，渐变由左上角到右下角线性进行，语法如下：

```
context.createLinearGradient(xStart, yStart, xEnd, yEnd)
```

- xStart，yStart：渐变的起始点的坐标
- xEnd，yEnd：渐变的结束点的坐标

代码第9行~第11行，调用线性渐变对象gradient的addColorStop方法，分别在开始点、中

间点、结束点添加对应颜色，语法如下：

```
context.createLinearGradient(xStart, yStart, xEnd, yEnd)
```

- offset：表示渐变的开始点和结束点之间的一部分，是范围在0.0~1.0之间的浮点值。
- color：表示指定offset显示的颜色。



Canvas渐变除了有线性渐变外还有径向渐变（关键API为createRadialGradient），读者有兴趣可以尝试修改本示例，熟悉两种渐变方法的使用。



6.4 示例4 输出图片文件

6.4.1 示例效果

回顾示例3，已经具备了添加图片、文字、渐变色功能。本例在其基础上，加入图片导出功能，同时还可以选择导出图片的类型和名称。

使用Chrome浏览器打开网页文件，运行效果如图6.9所示。

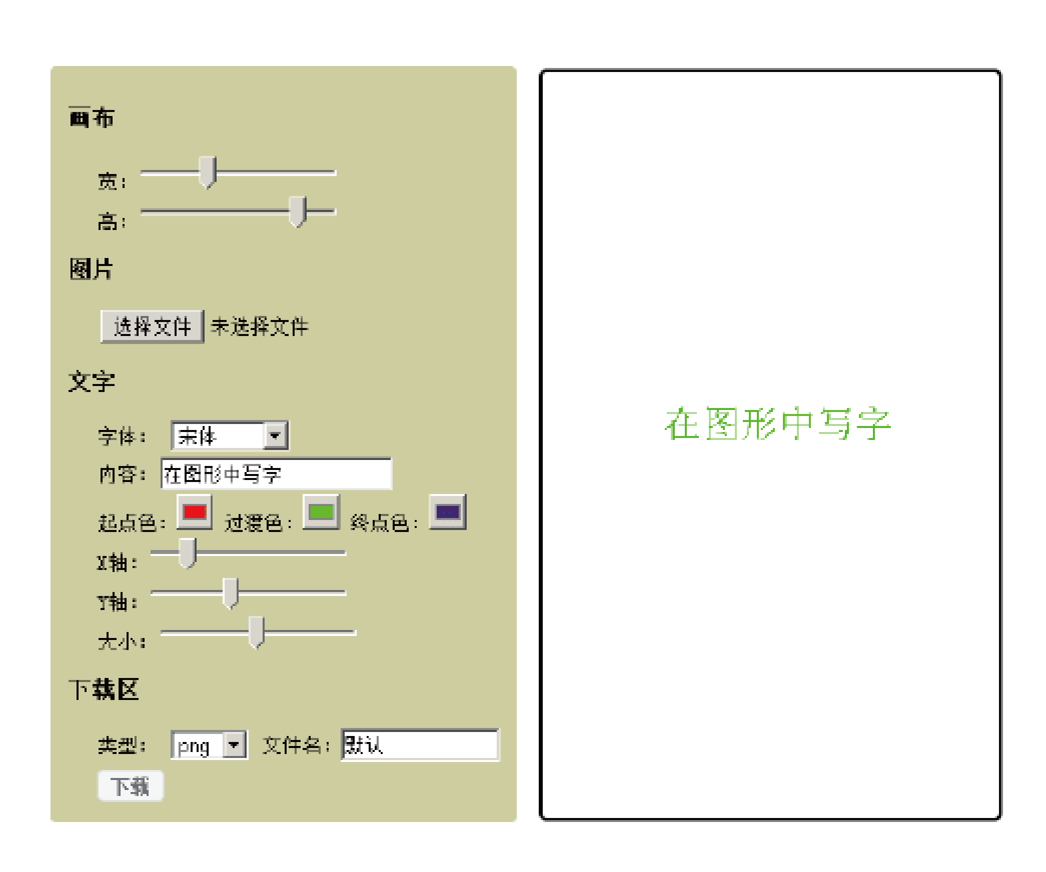
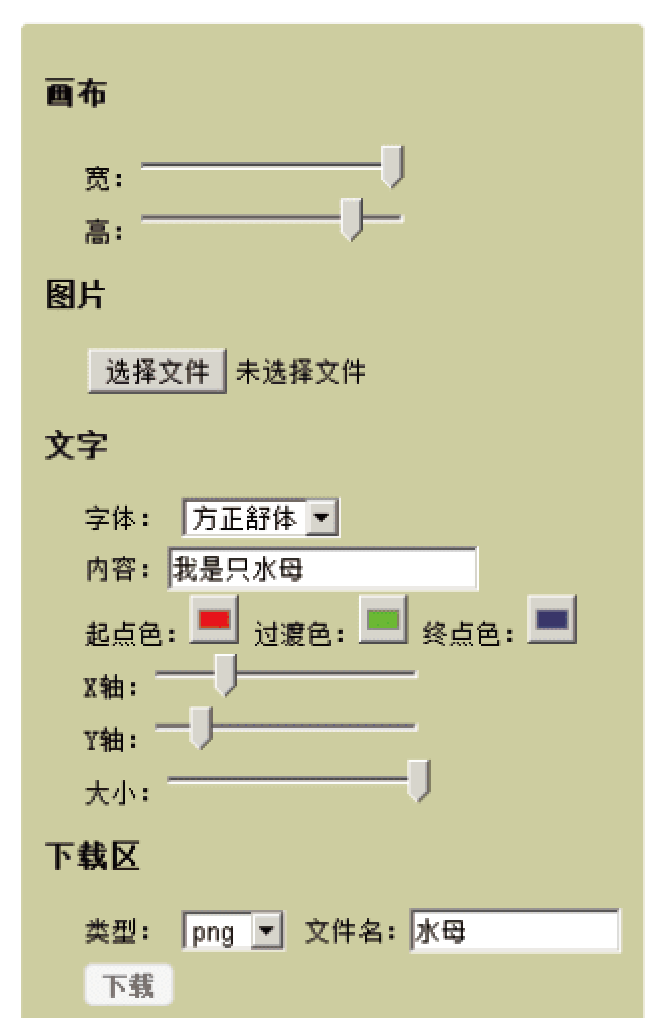


图6.9 使用Chrome打开网页文件

在画框的左侧调节框进行如下操作：

- 选择图片库中的“水母.jpg”图片。
- 调整画布的宽高。
- 调整文字字体为“方正舒体”。
- 修改图片文字内容为“我是只水母”。
- 调整文字在画框中的X轴和Y轴距离。
- 修改下载区的文件名为“水母”。

运行效果如图6.10所示。



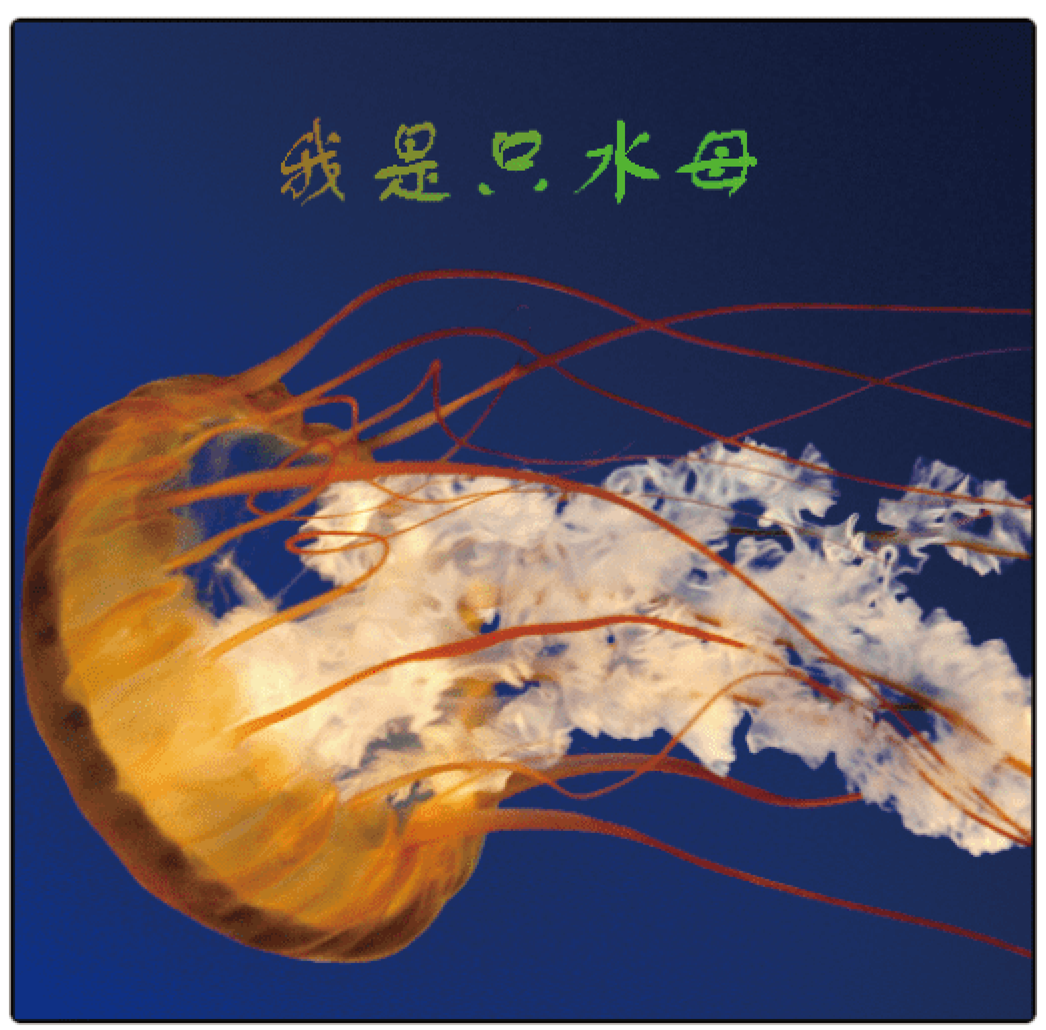


图6.10 选择图片并调整相关设置

单击“下载”按钮，浏览器出现“另存为”对话框，如图6.11所示。

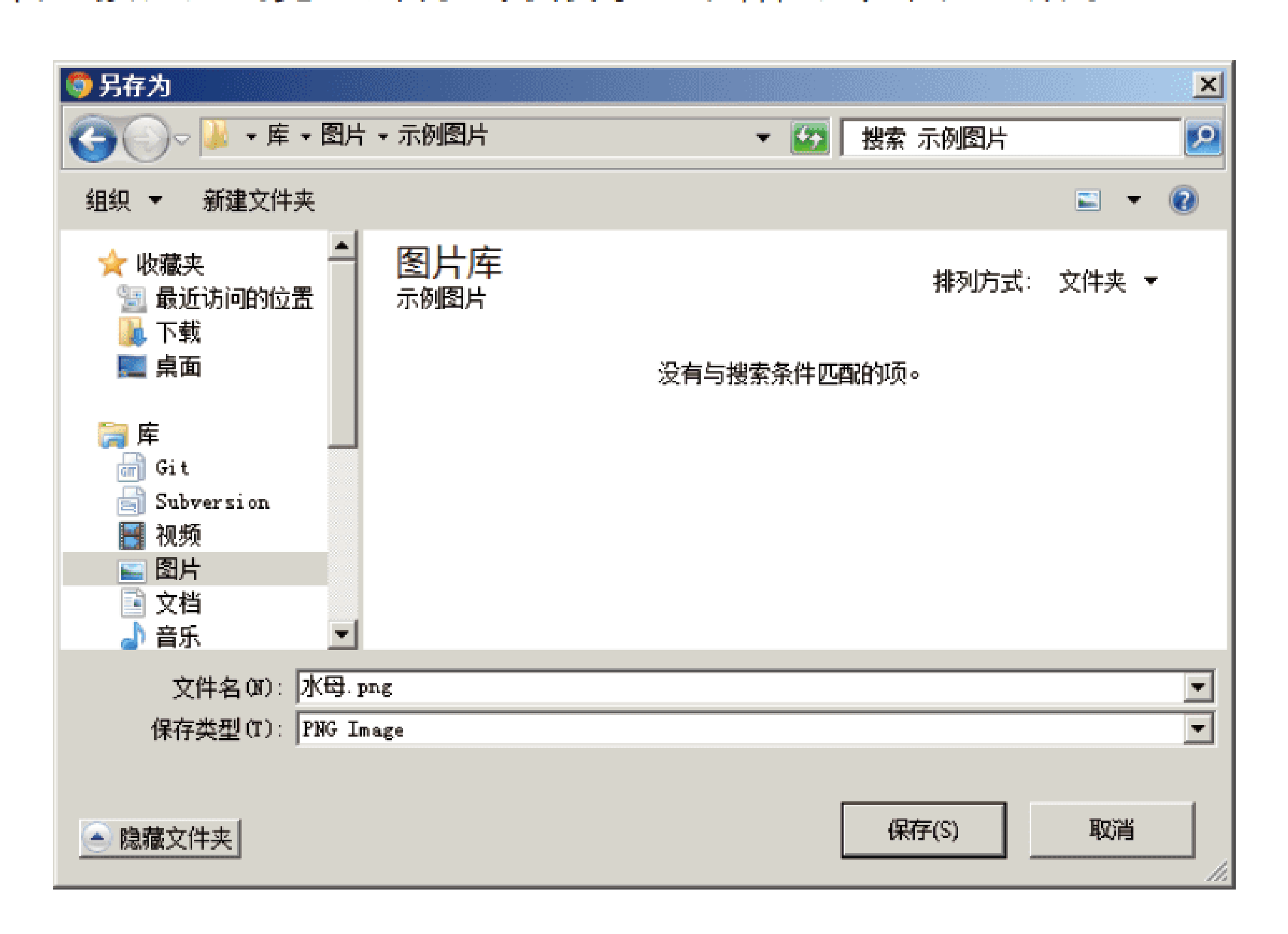


图6.11 “另存为”对话框



读者可以通过修改“类型”下拉框的值，调整下载图片类型。本例一共提供了png、jpeg、bmp、gif这4种类型。

6.4.2 代码分析

本例代码基本与示例3相同，下面就不同部分做一个分析。

HTML文档增加了33个元素，代码如下：

```
类型：<select id="img_type">
    <option value="png">png</option>
    <option value="jpeg">jpeg</option>
    <option value="bmp">bmp</option>
    <option value="gif">gif</option>
</select>
<input id="file_name" value="默认" type="text" />
<a class="download" id="btn_download" href="javascript:;">下载</a>
```

在draw函数的末尾增加两行代码，代码如下：

```
btn_download.setAttribute('href', canvas.toDataURL('image/' + img_type.
  value));
btn_download.setAttribute('download', file_name.value);
```

toDataURL方法，返回一张使用canvas绘制的符合data:URL格式图片。该方法接收一个type参数，表示为MIME类型，如image/png、image/jpeg等。如果是image/jpeg，可以有第二个参数，表示JPEG的质量等级，范围在0.0~1.0之间。



toDataURL方法绘制的图片格式在各个浏览器中的支持情况不同。标准草案可以参考网站http://www.w3.org/TR/2011/WD-html5-20110405/the-canvas-element.html#dom-canvas-todataurl。

第一行代码，将获取的图片值赋予下载按钮的href属性，用于给下载提供图片内容。

第二行代码，设置A元素的download属性。download属性在HTML 5的草案中被添加，用于表明一个资源是让用户下载的，并且下载文件与download属性值相同。

另外增加了对图片类型下拉框change事件的监听、对文件名输入框keyup事件监听，代码如下：

```
img_type.addEventListener('change', function () {// 监听图片下拉框change事件
    btn_download.setAttribute('href', canvas.toDataURL('image/' + img_
type.value));
});
file_name.addEventListener('keyup', function (e) {      // 监听文件名文本框按钮松开事件
    btn_download.setAttribute('download', file_name.value);
   // 设置下载A元素的download属性
});
```



6.5 示例5 操作图片像素

6.5.1 示例效果

如果用过Photoshop，会发现里面有多种针对图片的特效，本示例将通过Canvas的接口实现其中的两个效果，黑白和逆色。本例的基础功能建立在示例4的基础之上。

使用Chrome浏览器打开网页文件，运行效果如图6.12所示。

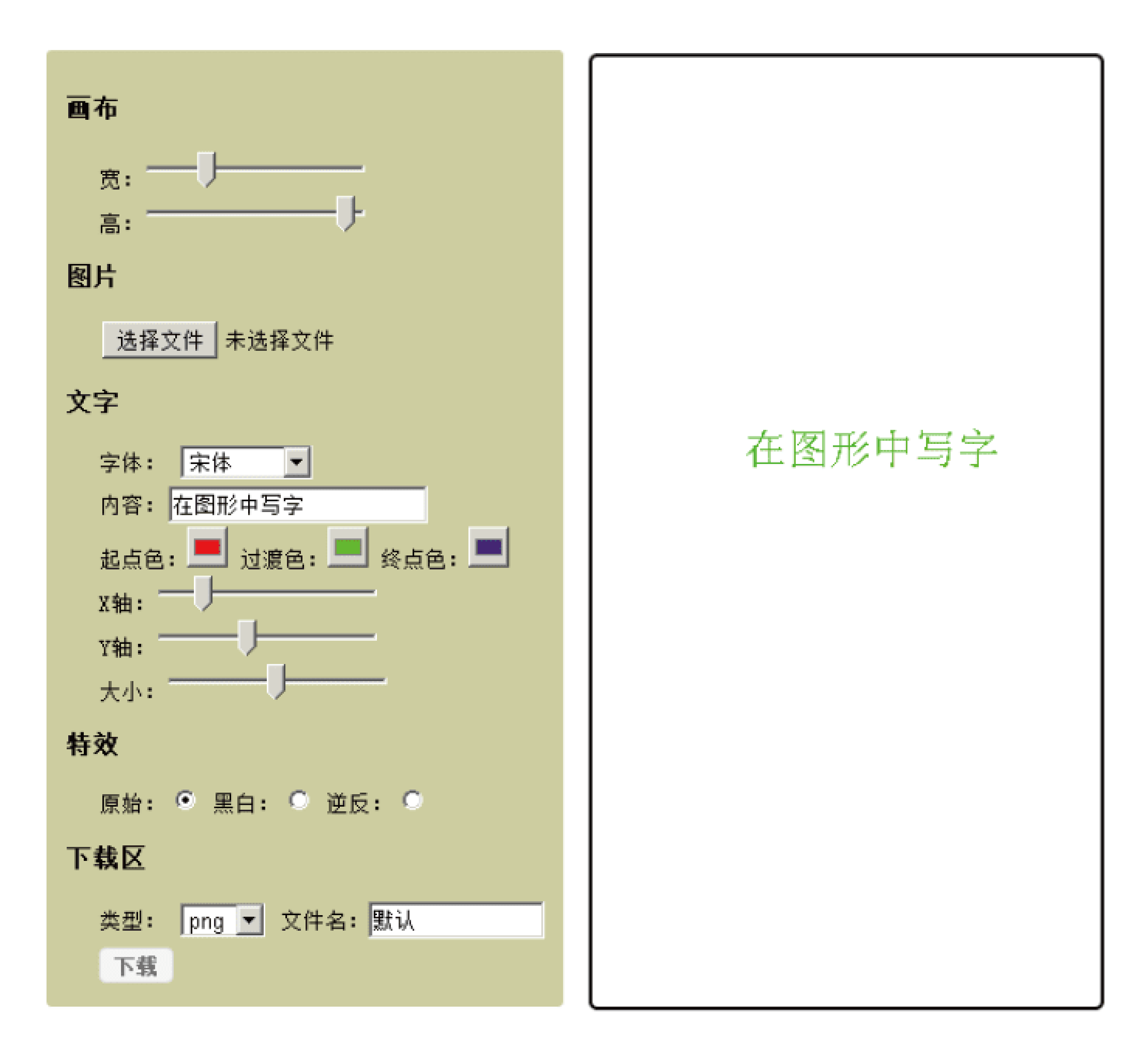


图6.12 使用Chrome打开网页文件

选择图片库的“菊花.jpg”图片，并对“画布”、“文字”调节区进行设置，“特效”区选择默认的“原始”，运行效果如图6.13所示。

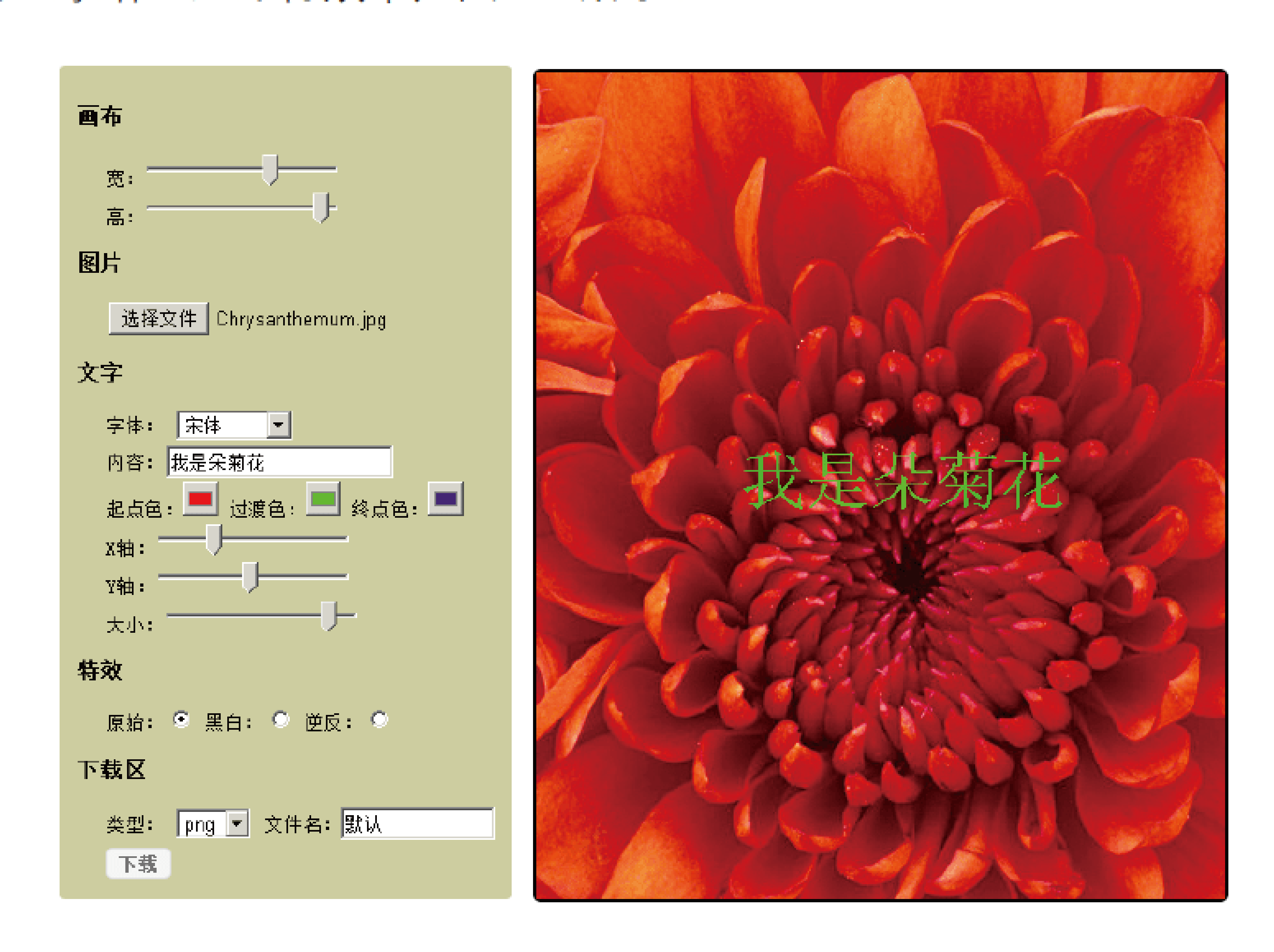


图6.13 选择图片库的“菊花.jpg”并调整相关设置

选中“黑白”单选项，右侧画布内容变为黑白色，效果如图6.14所示。



图6.14 选中“黑色”单选项



读者可以选中“逆反”单选项观察逆反色的效果。

6.5.2 代码分析

本例代码基本与示例4相同，下面就不同部分做一个分析。

HTML文档增加了“特效”相关内容，代码如下：

```
<li><h3>特效</h3></li>
<li class="item">
    <input  type="hidden"  id="effect_hidden"  value="Normal" />
    原始：<input  type="radio"  checked  name="effect"  value="Normal" />
    黑白：<input  type="radio"  name="effect"  value="BlackAndWhite" />
    逆反：<input  type="radio"  name="effect"  value="Antagonistic" />
</li>
```

隐藏输入框（type"hidden"）用于暂存每次单选项变化后选中的值，默认为Normal。

在左侧画框上有三个name属性值为effect的单选项。此处的name均相同，表示之间有且只能选择其中一个。每个单选项的值都设置成对应效果的英文名。

当用户选择不同的单选项时，触发绑定在change上的事件。回调函数先将获取选中单选项的值赋予隐藏输入框，然后对右侧画布进行重新绘制，代码如下：

```
var slice = Array.prototype.slice;                // 获取数组slice原型方法
var effects = slice.call(document.querySelectorAll('input[name=effect]'),
 0);     // 将获取的元素列表转为数组
effects.forEach(function (effect) {               // 循环元素数组
   // 每当数值变化触发回调，修改隐藏输入框的值，并重绘画布
    effect.addEventListener('change', function () {
        effect_hidden.value = this.value;         // 设置隐藏输入框的值
```

```
        draw();                                      // 重新绘制画布
    });
});
```



上面代码用到document的querySelectorAll方法，方法在W3C Selectors API Level 1规范中定义，用来返回所有满足选择器条件的元素列表。CSS选择器的规范可以参考网站http://www.w3.org/TR/css3-selectors/#selectors。

在介绍用draw函数增加内容之前，先看一组效果方法的函数集，代码如下：

```
01 var EFFECT = {
02     'Normal': function (image) { },                    // 原始效果
03     'BlackAndWhite': function (image) {                // 黑白效果
04         var data = image.data;           // 获取画布像素数据的data属性
05         for (var i = 0, l = data.length; i < l; i += 4) {
06             data[i] = data[i + 1] = data[i + 2] = parseInt((data[i]
   + data[i + 1] + data[i + 2]) / 3);
07         };
08         context.putImageData(image, 0, 0);     // 将改变后图像数据放回画布
09     },
10     'Antagonistic': function (image) {         // 逆色效果
11         var data = image.data;                 // 获取画布像素数据的data属性
12         for (var i = 0, l = data.length; i < l; i += 4) {
13             data[i] = 255 - data[i];
14             data[i + 1] = 255 - data[i + 1]
15             data[i + 2] = 255 - data[i + 2];
16         };
17         context.putImageData(image, 0, 0);     // 将改变后图像数据放回画布
18     }
19 };
```

Normal、BlackAndWhite、Antagonistic这三个方法均接受一个image参数，该参数表示像素数据，数据类型为ImageData对象。image.data表示像素的一维数组，数组的每个元素都分布在0~255之间。每4个元素组成一个像素，分别表示红、绿、蓝、Alpha。

Normal是一个空函数，不做任何事情。

BlackAndWhite将图片转换为黑白色。黑白色转换逻辑见上方代码第5~7行，以每4个元素为一节点循环像素数组。获取每次循环节点的前三个数组元素，将其值重新设置为三个元素的平均值。当循环结束后，调用putImageData方法将图像数据放回画布，putImageData方法语法如下：

```
context.putImageData(imgData, x, y, dirtyX, dirtyY, dirtyWidth,
 dirtyHeight)
```

- imgData：规定要放回画布的ImageData对象。
- x：ImageData对象左上角的x坐标，以像素计。
- y：ImageData对象左上角的y坐标，以像素计。
- dirtyX：可选，水平值x，以像素计，在画布上放置图像的位置。
- dirtyY：可选，水平值y，以像素计，在画布上放置图像的位置。

- dirtyWidth：可选，在画布上绘制图像所使用的宽度。
- dirtyHeight：可选，在画布上绘制图像所使用的高度。

Antagonistic将图片转换为逆色（即反色）。代码逻辑见上方代码第12行~第16行，算法是用255减去对应的红、绿、蓝值，达到反色效果。

最后介绍draw函数增加的部分（只有一行代码），代码如下：

```
EFFECT[effect_hidden.value].call(context, context.getImageData(0,
0, canvas.width, canvas.height))
```

该行代码出现在画布绘制文字与图片转Base64数据之间。代码放于该位置，是为了保证图片特效能作用于画布上的文字，同时绘制完的数据能同步更新到下载链接。代码用到getImageData方法，语法如下：

```
imgData=context.getImageData(x, y, width, height)
```

- x：开始复制的左上角位置的x坐标。
- y：开始复制的左上角位置的y坐标。
- width：将要复制的矩形区域的宽度。
- height：将要复制的矩形区域的高度。



6.6 示例6 制作动画计时器

6.6.1 示例效果

本例会运用前面示例介绍的Canvas接口，并使用CSS 3样式制作一个带有“开始”、“暂停”、“复位”功能的精美计时器，计时器最小精度为毫秒。

使用Chrome浏览器打开网页文件，运行效果如图6.15所示。



图6.15 使用Chrome打开网页文件

单击“开始”按钮，计时器开始计时，毫秒数开始疯狂跳动，运行效果如图6.16所示。



图6.16 使用Chrome打开网页文件

单击“暂停”按钮，计时器停止跑动。单击“开始”按钮，计时器重新从暂停位置开始继续计数。单击“复位”按钮，计数器清零。这里不多做图示，读者可以自己动手操作。

6.6.2 代码设计

利用编辑器打开“006.制作一个动画计时器.html”文件，代码如下：

```
<!DOCTYPE HTML>
<html>
<head>
    <style>
        div.wraper_i                                      /* 计时器背景底色 */
        {
            border: 1px solid #cccc99;
            border-radius: 10px;                          /* 圆角效果 */
            -moz-border-radius: 10px;
            -webkit-border-radius: 10px;
            background-color: green;
            width: 333px;
            padding: 10px;
        }
        div.wraper_ii                                     /* 计时器显示板底色 */
        {
            border: 1px solid #cccc99;
            border-radius: 10px;                          /* 圆角效果 */
            -moz-border-radius: 10px;
            -webkit-border-radius: 10px;
            background-color: #cccc99;
            padding: 5px;
        }
        div.buttons                                       /* 按钮外框容器样式 */
        {
            height: 30px;
            margin: 0 auto;
            padding: 8px 0 5px 30px;
        }
        ul.viewport                                       /* 计数显示屏样式 */
        {
            border: 1px solid #cccc99;
            border-radius: 10px;                          /* 圆角效果 */
            -moz-border-radius: 10px;
            -webkit-border-radius: 10px;
            background-color: green;
            height: 55px;
            padding: 5px 0 0 5px;
            margin: 0;
        }
        a.button                                          /* 按钮通用样式 */
        {
            display: block;
```

```
            float: left;
            position: relative;                /* 相对定位 */
            height: 25px
            width: 80px;
            margin: 0 10px 18px 0;
            text-decoration: none;             /* 去除相关文本修饰 */
            font: 12px Arial;
            font-weight: bold;
            line-height: 25px;
            text-align: center;
            -webkit-border-radius: 3px;        /* 圆角效果 */
            -moz-border-radius: 3px;
            border-radius: 3px;
        }
        a.button:before,
        a.button:after                  /* 在按钮元素之前和之后添加样式 */
        {
            content: '';                       /* 在元素上添加内容 */
            position: absolute;                /* 绝对定位 */
            left: -1px;
            height: 25px;
            width: 80px;
            bottom: -1px;
            -webkit-border-radius: 3px;        /* 圆角效果 */
            -moz-border-radius: 3px;
            border-radius: 3px;
        }
        a.button:before                        /* 在按钮元素之后添加 */
        {
            height: 23px;
            bottom: -4px;
            border-top: 0;
            border-radius: 0 0 3px 3px;        /* 圆角效果 */
            -webkit-border-radius: 0 0 3px 3px;
            -moz-border-radius: 0 0 3px 3px;
            box-shadow: 0 1px 1px 0px #bfbfbf;        /* 图层阴影效果 */
            -webkit-box-shadow: 0 1px 1px 0px #bfbfbf;
            -moz-box-shadow: 0 1px 1px 0px #bfbfbf;
        }
        a.button:active          /* 元素激活，单击与释放之间样式 */
        {
            border: none;
            bottom: -4px;
            margin-bottom: 22px;
            -webkit-box-shadow: 0 1px 1px #fff;
            -moz-box-shadow: 0 1px 1px #fff;
            /* 添加内阴影，让按钮看上去感觉已被按下 */
            box-shadow: 1px 1px 0 #fff, inset 0 1px 1px rgba(0, 0, 0, 0.3);
        }
        a.button:active:before,
        a.button:active:after           /* 元素激活之前和之后添加样式 */
```

```
        {
            border: none;
            -webkit-box-shadow: none;          /* 去除图层阴影效果 */
            -moz-box-shadow: none;
            box-shadow: none;
        }
        a.green,
        a.green:hover,
        a.green:visited          /* 绿色按钮、悬停、或已被访问的链接 */
        {
            color: #5d7731;
            border-bottom: 4px solid #799545;
            text-shadow: 0px 1px 0px #d5e8aa;
            background: #cae285;
            /* 线性渐变 */
            background: -webkit-gradient(linear, left top, left bottom, from(#cae285), to(#a3cd5a));
            background: -moz-linear-gradient(top,  #cae285,  #a3cd5a);
            /* 内阴影 */
            box-shadow: inset 1px 1px 0 #cce3a1;
        }
        .green:before,
        .green:after             /* 在绿色按钮元素之前和之后添加样式 */
        {
            border: 1px solid #98b85b;
            border-bottom: 1px solid #6d883b;
        }
        a.green:hover
        {
            background: #a3cd5a;        /* 在绿色按钮悬停添加样式 */
            /* 线性渐变 */
            background: -webkit-gradient(linear, left top, left bottom, from(#a3cd5a), to(#cae285));
            background: -moz-linear-gradient(top,  #a3cd5a,  #cae285);
        }
    </style>
</head>
<body>
    <header><h2>制作一个动画计时器</h2></header>
    <section>
       <div class="wraper_i">
         <div class="wraper_ii">
            <ul class="viewport">
                <!-- 计时屏 -->
                <canvas width="315" height="50"></canvas>
            </ul>
            <!-- 操作区 -->
            <div class="buttons">
                <a href="#" class="button green J_start">开始</a>
                <a href="#" class="button green J_stop">暂停</a>
                <a href="#" class="button green J_reset">复位</a>
```

```
                </div>
            </div>
        </div>
    </section>
</body>
<script>
    var canvas = document.querySelector('canvas'),    // 获取Canvas画布元素
        context = canvas.getContext("2d"),    // 获取Canvas元素上下文
        start = document.querySelector('a.J_start'),        // 开始按钮
        stop = document.querySelector('a.J_stop'),          // 暂停按钮
        reset = document.querySelector('a.J_reset');        // 复位按钮
    function draw(text) {                                  // 绘制计时器数字
        // 清空画布指定矩形区域内容
        context.clearRect(0, 0, parseInt(canvas.width),
parseInt(canvas.height));
        context.fillStyle = 'black';               // 画布计时器文字颜色为黑色
        context.font = 'italic 50px sans-serif';      // 设置文字字体和大小
        context.textBaseline = 'top';                 // 设置文本基线
        context.fillText(text || '00:00:00:000', 0, 0);
        // 填充文字到画布指定区域
    };
    var start_time,                                // 单击开始时间
        time_spend,                                // 时差（毫秒）
        time_stop = 0,                             // 停顿时的时差（毫秒）
        interval;                                  // 轮询变量
    function runtime() {
     // 计算时差，返回指定格式时间hh:MM:ss:SSS
       time_spend = (new Date() - start_time + time_stop) || 0;
        // 时差，默认为0
       var hour,  minute,  second,  millisecond,
           temp_time_spend = time_spend;       // 临时变量，存放的是差值
        millisecond = temp_time_spend % 1000;     // 取1000的模获取毫秒
        temp_time_spend = parseInt(temp_time_spend / 1000);
         // 获取剔除毫秒后的秒
        second = temp_time_spend % 60;         // 取60的模获取秒
        temp_time_spend = (temp_time_spend - second) / 60;
         // 获取剔除秒后的分
        minute = temp_time_spend % 60;                // 取60的模获取分
        hour = (temp_time_spend - minute) / 60;       // 获取剔除分后的小时
        second = second < 10 ? ('0' + second) : second;     // 不足2位补0
        minute = minute < 10 ? ('0' + minute) : minute;
        hour = hour < 10 ? ('0' + hour) : hour;
        return [hour, minute, second, millisecond].join(':');
    };
    start.addEventListener('click', function (e) {    // 开始按钮单击事件
        e.preventDefault();                           // 禁止a链接默认事件
        if (!interval) {
            start_time = new Date();
            interval = setInterval(function () {      // 每10ms执行一次
                draw(runtime());                      // 获取对应的计数并绘制
            }, 10);
```

```
189         };
190     }, false);
191     stop.addEventListener('click', function (e) {     // 暂停按钮单击事件
192         e.preventDefault();
193         time_stop = time_spend;                       // 缓存停止时候的时差
194         clearInterval(interval);
195         interval = null;
196     }, false);
197     reset.addEventListener('click', function (e) {    // 复位按钮单击事件
198         e.preventDefault();
199         time_stop = 0;                                // 清零时差
200         clearInterval(interval);
201         interval = null;
202         draw();                                       // 绘制默认计数
203     }, false);
204     draw();
205 </script>
206 </html>
```

6.6.3 代码分析

1. 分析样式

代码第5行~第40行，分别定义了外框背景、计时器背景板、计时器视窗、按钮面板4大样式类。均运用了CSS 3的圆角效果，样式关键字：border-radius。

代码第 41行~第127行，定义了按钮的基础样式类和绿色样式类，运用了多种伪类效果。CSS中的伪类主要用于向某些选择器添加特殊的效果，下面罗列了示例中出现的伪类。

- :before：在元素内容的最前面插入生成内容。
- :after：在元素内容的最后面插入生成内容。
- :active：激活状态的元素添加样式，如单击按钮与释放之间。
- :active:before：元素被激活前插入生成内容。
- :active:after：元素被激活后插入生成内容。
- :hover：当鼠标悬停在元素上。
- :visited：已经被访问的链接。



除了文中提到的CSS伪类以外，还有大量的其他类型伪类在日常的开发中被使用。CSS伪类的应用极大地简化了代码逻辑的复杂度，相信CSS未来的版本会继续对伪类进行加强。想了解更多的伪类使用，可以参考网站http://www.w3schools.com/css/css_pseudo_classes.asp。

2. 分析脚本代码逻辑

代码第150行~第154行，从文档中获取需要操作的元素。

代码第155行~第162行，在画布上绘制传入的字符串，这块代码在前面的示例中已经多次出现，如果有疑惑可以参考代码后方注释。

代码第163行~第166行，声明用于计数相关的中间变量。

代码第167行~第181行，定义了关键的计时函数runtime。第168行代码计算计时器跑过的时间，单位以毫秒计。原理是将当前时间减去单击“开始”时刻的时间，然后加上被单击“暂停”按钮之前跑过的时间。函数的后面部分是将得到的总毫秒数进行格式转换，返回一个类似于“小时：分钟：秒：毫秒”的字符串。

代码第182行~第190行，给“开始”按钮的单击事件绑定监听。当用户单击按钮时，先判断计数器是否已经启动。如果未启动，则记录当前单击的时间，并以10ms为一周期执行runtime函数并绘制结果。

代码第191行~第196行，给“暂停”按钮的单击事件绑定监听。当用户单击按钮时，用变量time_stop记录计数器当前走过的时间，然后停止周期调用函数。

代码第197行~第203行，给“复位”按钮的单击事件绑定监听。当用户单击按钮时，重置所有变量，并重新绘制默认计数器视图。



示例7 在画布中剪贴图像

6.7.1 示例效果

在很多社交网站上能看到这样的场景，用户上传自定义头像，并对其进行编辑保存。多数网站都实现了图像编辑功能，采用的是Flash技术。本例将采用HTML 5技术实现此场景，具备上传图片和简单的剪贴功能。

使用Chrome浏览器打开网页文件，运行效果如图6.17所示。

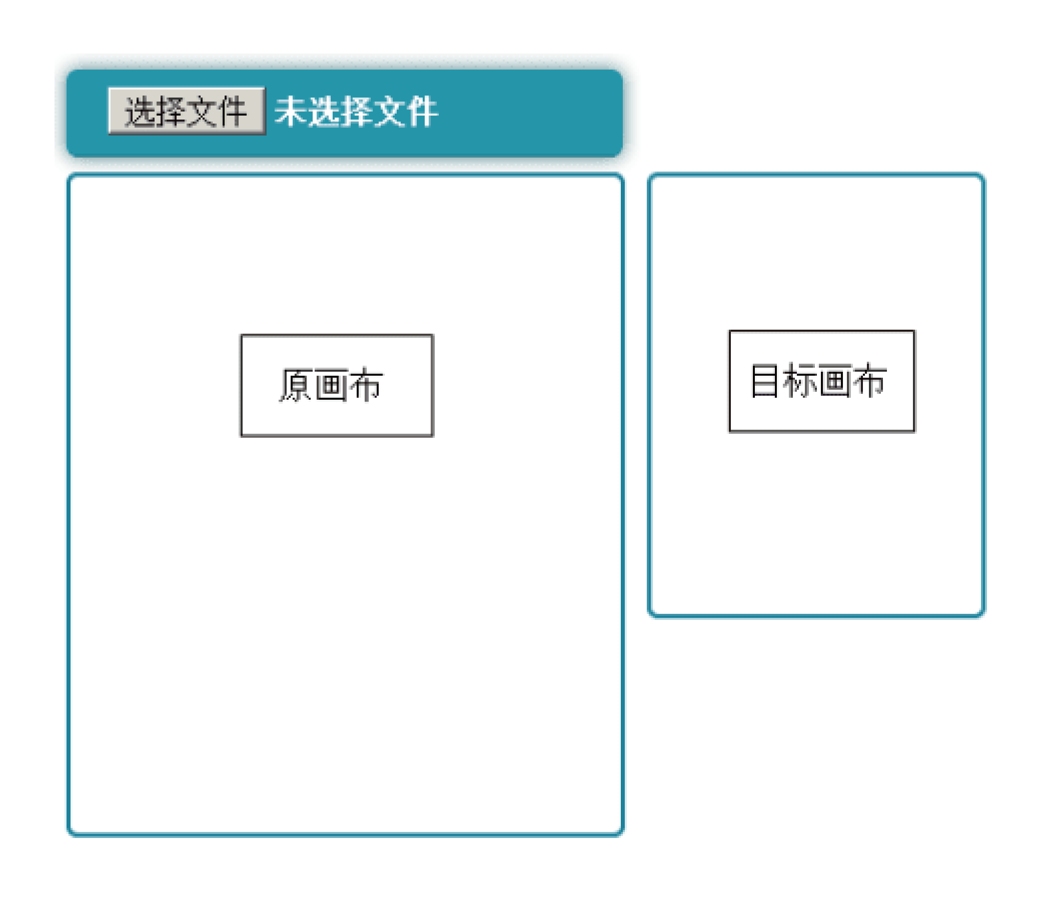


图6.17 使用Chrome打开网页文件

单击“选择文件”按钮，在图库中选择Einstein.png图片并打开，效果如图6.18所示。

在人物的头部左上方按住鼠标左键，并沿着右下方拖动，出现一个虚线框。虚线区域中

将被剪贴到右侧目标画布中，效果如图6.19所示。松开鼠标左键，原画布虚线框内的爱因斯坦头像被复制到右侧目标画布，效果如图6.20所示。

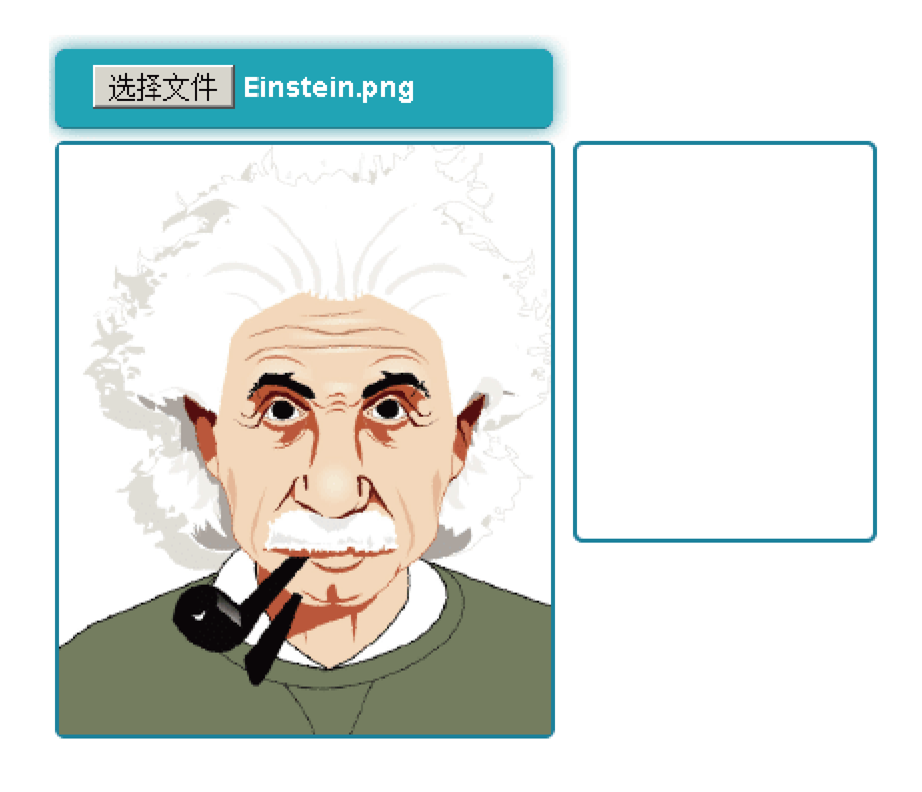


图6.18　单击“选择文件”按钮并打开图片

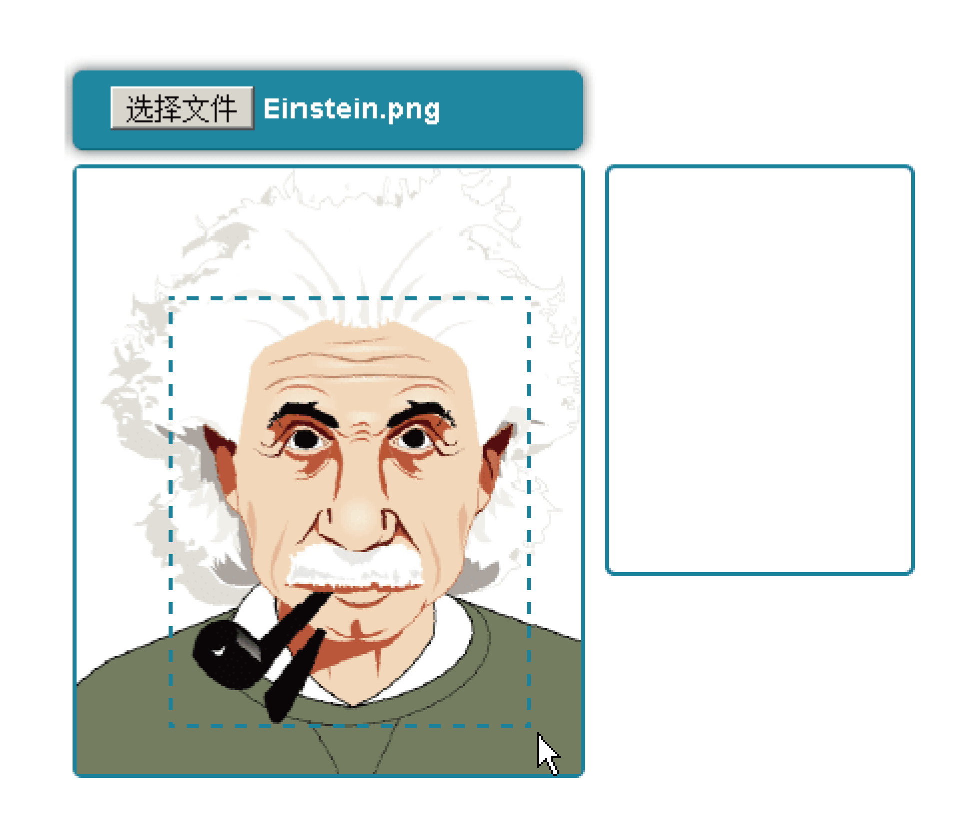
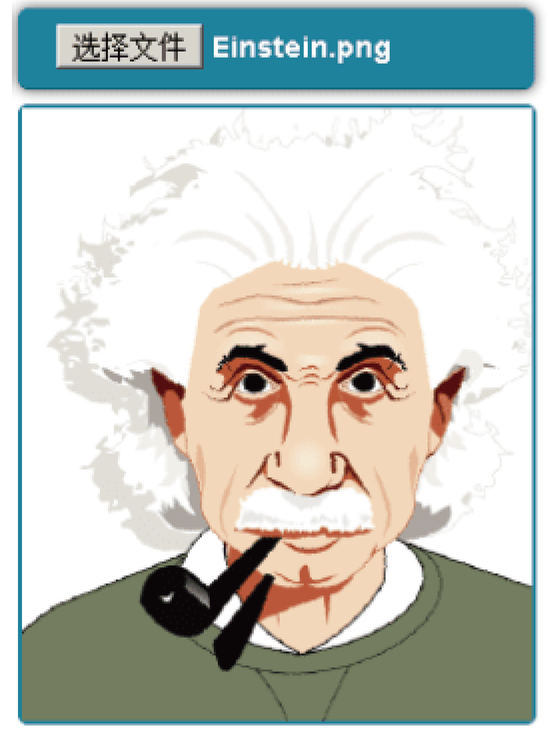


图6.19　单击并拖动



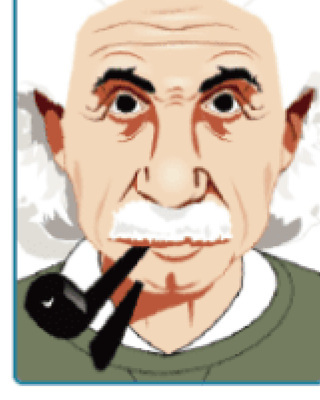


图6.20　松开鼠标左键，头像被截取

6.7.2 代码设计

利用编辑器打开“007.在画布中剪贴图像.html”文件，代码如下：

```
<!DOCTYPE HTML>
<html>
<head>
    <style>
    @-webkit-keyframes bluePulse {                    /* 蓝色闪动动画 */
            from { background-color: #007d9a; -webkit-box-shadow: 0 0 9px #333; }
            50% { background-color: #2daebf; -webkit-box-shadow: 0 0 18px #2daebf; }
            to { background-color: #007d9a; -webkit-box-shadow: 0 0 9px #333; }
        }
```

```
    .button {                                       /* 上传按钮样式 */
        // ......省略部分样式代码
        -webkit-animation-name: bluePulse;     /* 设置执行动画名称 */
        -webkit-animation-duration: 2s;        /* 动画一周期的时间 */
        -webkit-animation-iteration-count: infinite;   /* 无限次的循环播
放动画 */
     }
      // ...... 此处省略部分样式代码，请参考光盘源代码
    </style>
</head>
<body>
    <header><h2>在画布中剪贴图像</h2></header>
    <section>
        <!-- 图片按钮上传 -->
        <input type="file" class="button"/>
        <!-- 原画布 -->
        <canvas id="J_canvas_i" width="250" height="300"></canvas>
        <!-- 目标画布 -->
        <canvas id="J_canvas_ii" width="150" height="200"></canvas>
    </section>
</body>
<script>
    var canvas_i = document.getElementById('J_canvas_i'),
    // 获取Canvas画布元素
        context_i = canvas_i.getContext("2d"),
        // 获取Canvas元素上下文
        canvas_ii = document.getElementById('J_canvas_ii'),
        // 获取目标Canvas画布元素
        context_ii = canvas_ii.getContext("2d"),
        // 获取目标Canvas元素上下文
        input_file = document.querySelector('input[type=file]'),
        // 按上传按钮
        clip_wraper = document.createElement('div'),
        // 自创建剪贴框元素
        img = new Image();                     // 新建图片元素实例
    clip_wraper.setAttribute('class', 'clip');        // 设置剪贴框样式
    function draw() {                                 // 绘制上传图片
        // 清空原画布指定矩形区域内容
        context_i.clearRect(0, 0, parseInt(canvas_i.width),
parseInt(canvas_i.height));
        // 清空目标画布指定矩形区域内容
        context_ii.clearRect(0, 0, parseInt(canvas_ii.width),
 parseInt(canvas_ii.height));
        context_i.drawImage(img, 0, 0);               // 填充图片到画布
    };
    //获取或设置元素样式。参数1为目标元素；参数2如果为字符则获取样式，如果为对象
则设置元素样式
    function css(element, options) {
        if (typeof options === 'string') {    return element.
style[options];
        } else {
```

```
            for (var name in options) {element.style[name] = options[name];  };
        };
    };
    img.addEventListener('load', draw, false);
    // 监听图片的加载完毕事件
    input_file.addEventListener('change', function () {
    // 上传按钮值改变事件
        // ...... 此处省略部分代码，请参考光盘源代码
      }, false);
    var start_x = 0, start_y = 0,               // 单击的X轴和Y轴坐标位置
        move_x = 0, move_y = 0,                 // 单击后移动的相对距离
        offset_xy = 6;                          // 剪贴框距鼠标的相对位置
    function canvas_mousemove(e) {              // 鼠标移动事件函数
        // 鼠标移动时屏幕位置减去单击时位置获取鼠标移动的相对距离
        move_x = e.clientX - start_x;  move_y = e.clientY - start_y;
        if (move_x > 0 && move_y > 0) {         // 鼠标往右下方移动
            css(clip_wraper, {
                'width': move_x + 'px', 'height': move_y + 'px',
                'top': (start_y - offset_xy) + 'px', 'left': (start_x - offset_xy) + 'px'
         }
            // ......此处省略鼠标往右上、左下、左上移动的代码，请参考光盘源代码
    };
    // 监听剪贴框鼠标移动事件
    clip_wraper.addEventListener('mousemove', canvas_mousemove, false);
    canvas_i.addEventListener('mousedown', function (e) {
    // 监听原画布单击事件
        css(clip_wraper, {
            // 获取单击时的屏幕位置，设置剪贴框样式，并临时存放到变量中
            'left': (start_x = (e.clientX - offset_xy)) + 'px', 'top': (start_y = (e.clientY - offset_xy)) + 'px',
            'width': '1px', 'height': '1px'
        });
        document.body.appendChild(clip_wraper);
        // 将剪贴框添加到DOM文档内
        // 监听原画布的鼠标移动事件，绑定canvas_mousemove函数
        canvas_i.addEventListener('mousemove', canvas_mousemove, false);
    }, false);
    document.addEventListener('mouseup', function (e) {
    // 当鼠标在DOM文档内释放后触发;
        // 移除原画布的鼠标移动事件
        canvas_i.removeEventListener('mousemove', canvas_mousemove, false);
        if (e.target.nodeName.toLowerCase() == 'canvas' &&
            move_x > offset_xy && move_y > offset_xy &&
            // 移动的相对距离必须超过预设距离
            img.src.length > 0                  // 判断是否已经上传图片
            ) {
```

```
89              var sourceX = parseInt(css(clip_wraper, 'left')) - canvas_
   i.offsetLeft,
90                  sourceY = parseInt(css(clip_wraper, 'top')) - canvas_
    i.offsetTop,
91                  destWidth = parseInt(canvas_ii.width), destHeight =
    parseInt(canvas_ii.height);
92              context_ii.clearRect(0, 0, destWidth, destHeight);
               // 清空目标画布内容
93              // 填充图片到画布
94             context_ii.drawImage(img, sourceX, sourceY, Math.abs(move_
    x), Math.abs(move_y), 0, 0,
95 destWidth, destHeight);
96          };
97          try { document.body.removeChild(clip_wraper); } catch (e) { };
            // 从DOM文档中移除剪贴框
98      }, false);
99      // 阻止文档内容选择事件，避免拖动时触发内容选择造成不便
100     document.addEventListener('selectstart', function (e)
    { e.preventDefault() }, false);
101 </script>
102 </html>
```

6.7.3 代码分析

首先看代码第05行~第09行，使用了CSS 3的“@keyframes”创建了一个动画。示例中使用“@-webkit-keyframes”，表示只适用于WebKit内核的浏览器。“@-webkit-keyframes”后方紧跟的标识符表示动画的名称，大括号内为动画的过程。关键词from和to，等同于0%和100%，以上的代码等同于如下代码：

```
@-webkit-keyframes bluePulse {
         0% { background-color: #007d9a; -webkit-box-shadow: 0 0 9px #333;
    }
         50% { background-color: #2daebf; -webkit-box-shadow: 0 0 18px
    #2daebf; }
         100% { background-color: #007d9a; -webkit-box-shadow: 0 0 9px
    #333; }
}
```

样式类button中使用了定义的bluePulse动画，代码如下：

```
. button{
   ......
   -webkit-animation-name: bluePulse;
   -webkit-animation-duration: 2s;
   -webkit-animation-iteration-count: infinite;
}
```

- -webkit-animation-name：动画名。
- -webkit-animation-duration：完成一个周期所花费的秒数，默认是0。
- -webkit-animation-iteration-count：动画被播放的次数，默认是1，示例中infinite表示无限。



CSS 3动画还提供其他丰富的属性，想了解更多内容可以浏览网址http://www.w3school.com.cn/css3/css3_animations.asp。

接着分析脚本逻辑的关键部分，其余部分读者可以参考代码后方注释。

代码第39行~第45行定义函数draw，作用是在原画布绘制上传图片，并清空目标画布内容。

代码第47行~第52行定义函数css，封装了一个简单设置和获取元素样式的方法，没有过多的对浏览器兼容性做处理，读者可以使用第三方类库（如jQuery）替代该临时方法。

代码第57行~第59行定了5个变量，说明如下。

- start_x，start_y：记录用户单击时在浏览器上的坐标。
- move_x，move_y：记录单击后在原画布移动的相对距离。
- offset_xy：预设鼠标离虚线剪贴框的距离。

代码第60行~第69行定义函数canvas_mousemove，为虚线剪贴框运行拖动的核心函数。首先，计算获得单击后移动的相对距离，然后根据坐标轴的4个区间，分成左上、左下、右上、右下4个方向。通过判断移动距离的正负关系，即能得到鼠标的移动方向，最后设置虚线剪贴框的样式。

代码第72行~第81行，监听原画布的mousedown事件。用户在原画布单击后，在DOM文档内插入一个宽高为1px的DIV元素表示剪贴框，并添加对原画布mousemove事件的监听，监听函数为canvas_mousemove。

代码第82行~第98行监听DOM文档的mouseup事件。鼠标松开后，移除绑定在目标画布mousemove事件上的canvas_mousemove函数，防止多次绑定。通过if条件过滤后，计算矩形虚线剪贴框左上角相对于原画布的距离（像素计），然后调用目标画布drawImage方法，在目标画布上绘制剪贴图片。



drawImage方法使用可以参考本章示例2。



6.8 示例8 实现相片的360°旋转特效

6.8.1 示例效果

在示例7中，已经实现了截取上传图像指定区域的功能。本例将加入对截取后图像的360°旋转功能，用户可以轻松对图像进行4个方向的旋转。

使用Chrome浏览器打开网页文件，目标画布下方比示例7多出一个旋转按钮，运行效果如图6.21所示。

单击“选择文件”按钮，在图库中选择“Einstein-180.png”图片并打开，该图片是

Einstein.png的180°旋转图，效果如图6.22所示。使用鼠标截取原画布“爱因斯坦”的头部，此时原画布与目标画布图像均是倒立的，效果如图6.23所示。双击“旋转”按钮，进行两次90°目标图片旋转，效果如图6.24所示。

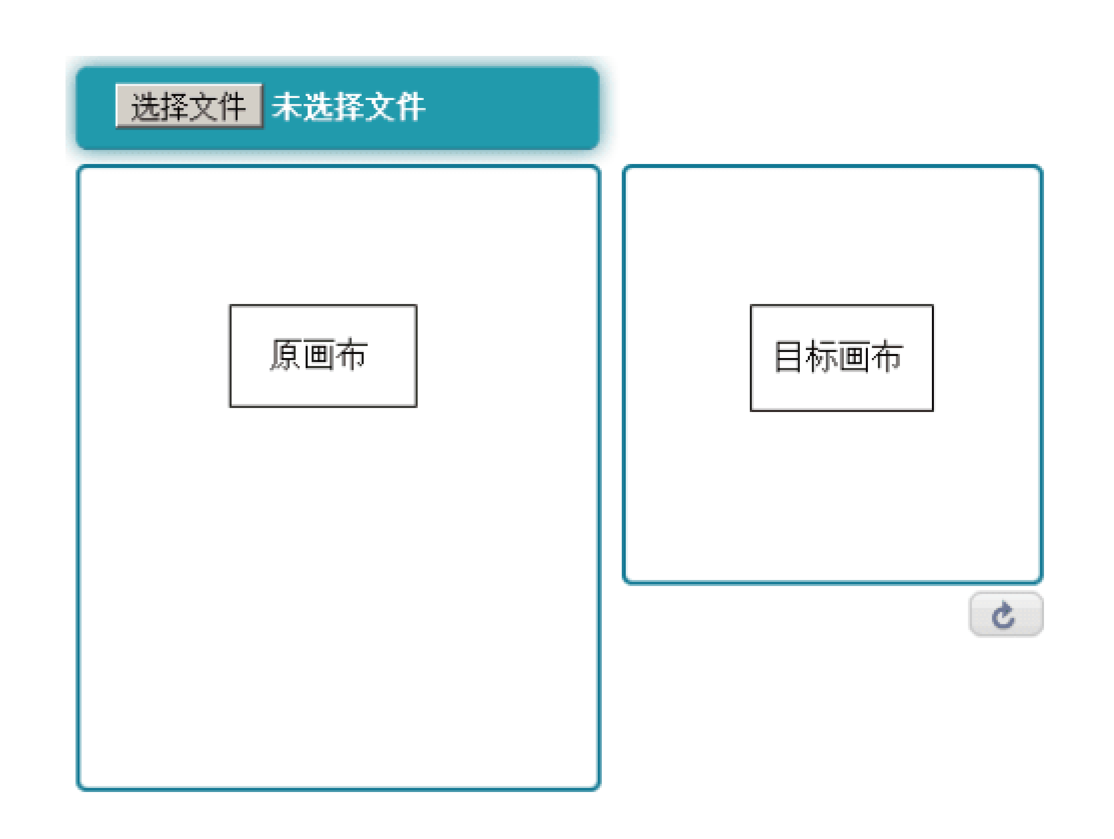


图6.21 使用Chrome打开网页文件



图6.22 单击“选择文件”按钮并打开图片

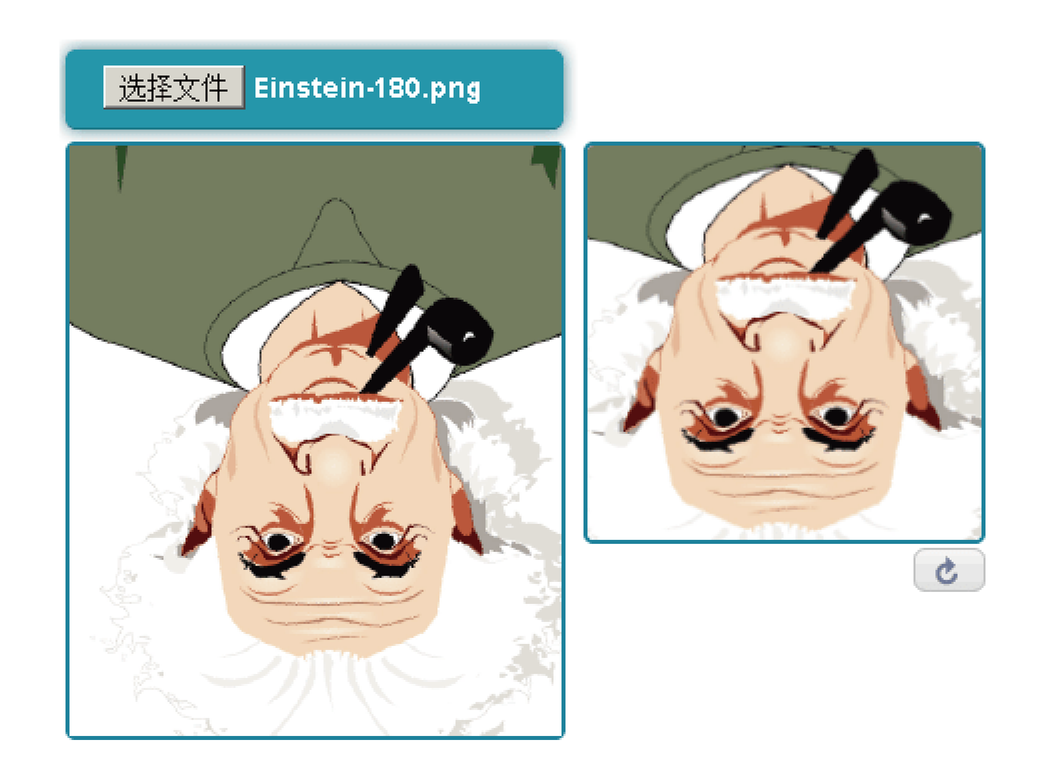


图6.23 单击“选择文件”按钮并打开图片

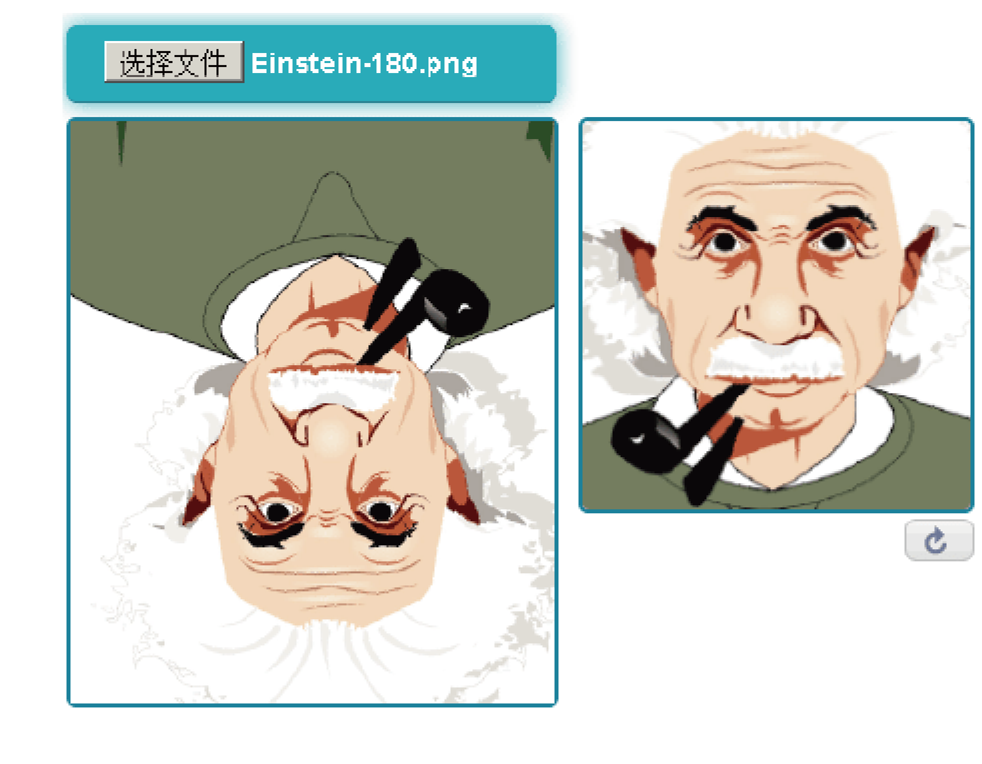


图6.24 双击“旋转”按钮

6.8.2 代码分析

本例代码多数与示例7相同，不同部分主要集中在处理图片旋转的脚本逻辑，下面就这部分做一个分析，代码如下：

```
var rotation_ii = document.getElementById('J_rotation_ii'),
    rotation_angle = [0, 90, 180, 270],        // 4个方向的变化角度
    temp_image = new Image();                  // 临时目标画框图片
rotation_ii.addEventListener('click', function () {
    if (!temp_image.src.length) {
        return;
    };
    var ANGLE_KEY_NAME = 'data-angleindex',    // 缓存在按钮元素上的属性名称
        // 获取当前索引数
        angle_index = parseInt(rotation_ii.getAttribute(ANGLE_KEY_NAME)) || 0,
```

```
11          width = parseInt(canvas_ii.width),
12          height = parseInt(canvas_ii.height),
13          drawX,
14          drawY;
15      angle_index++;                          // 计数加1，用于下轮单击
16      if (angle_index === rotation_angle.length) {     // 进入新的轮询
17          angle_index = 0;
18      };
19      switch (angle_index) {                         // 获取目标画框绘制的坐标
20          case 0: drawX = drawY = 0; break;      // 正位
21          case 1: drawX = 0; drawY = -height; break;    // 顺时针90°
22          case 2: drawX = -width; drawY = -height; break;      // 顺时针180°
23          case 3: drawX = -width; drawY = 0; break;     // 顺时针270°
24      };
25      context_ii.clearRect(0, 0, width, height);
26      context_ii.save();
27      context_ii.rotate(rotation_angle[angle_index] * Math.PI / 180);
28      context_ii.drawImage(temp_image, drawX, drawY);
29      context_ii.restore();
30      rotation_ii.setAttribute(ANGLE_KEY_NAME, angle_index);
        // 将计数写入按钮属性
31  });
```

代码第02行，定义变量rotation_angle，用于存储图片旋转的4个角度值。

代码第03行，定义变量temp_image，存放一个Image的实例。该图片实例的src属性会在每次截取图片完毕后被赋值，代码如下：：

```
document.addEventListener('mouseup', function (e) {
    // ......省略，详见本书网络资源
    temp_image.src = canvas_ii.toDataURL();
});
```

代码第05行~第07行，判断目标temp_image的src属性是否为空，即判断目标画布是否被绘制内容。如果没有被绘制，则不继续往下走。

在计算图片左上角位置之前，先了解Canvas的rotate方法，方法语法如下：

```
context.rotate(angle)
// angle表示旋转的量，用弧度表示。正值表示顺时针方向旋转，负值表示逆时针方向旋转
```

rotate方法旋转效果如图6.25所示。

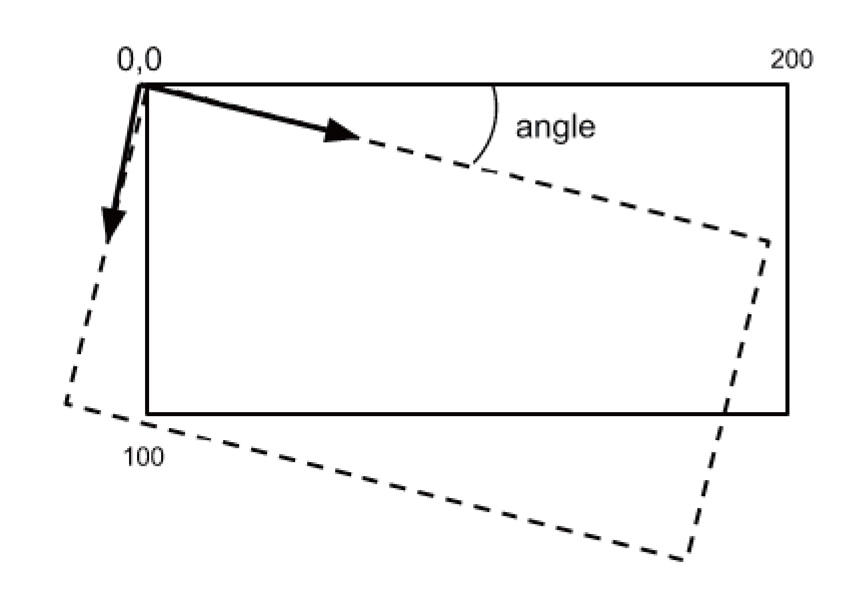


图6.25 rotate方法旋转效果

代码第19行~第24行给出了计算4种变化对应的图片左上角位置值。

在旋转图片的逻辑中，还需要特别注意以下两个关键方法。

- save：把当前状态的一份拷贝压入到一个保存图像状态的栈中。
- restore：从栈中弹出存储的图形状态并恢复画布对象的属性、剪切路径和变换矩阵的值。



6.9 示例9 一个HTML 5版销售数据图表

6.9.1 示例效果

数据可视化是将数据以图像的形式表现出来，图表在其中具有非常核心的地位。常见的图表有柱形图、折线图、饼图、条形图、雷达图等。本例采用柱形图方式展现销售数据信息。

使用Chrome浏览器打开网页文件，页面直接绘制出一张柱形图表，运行效果如图6.26所示。

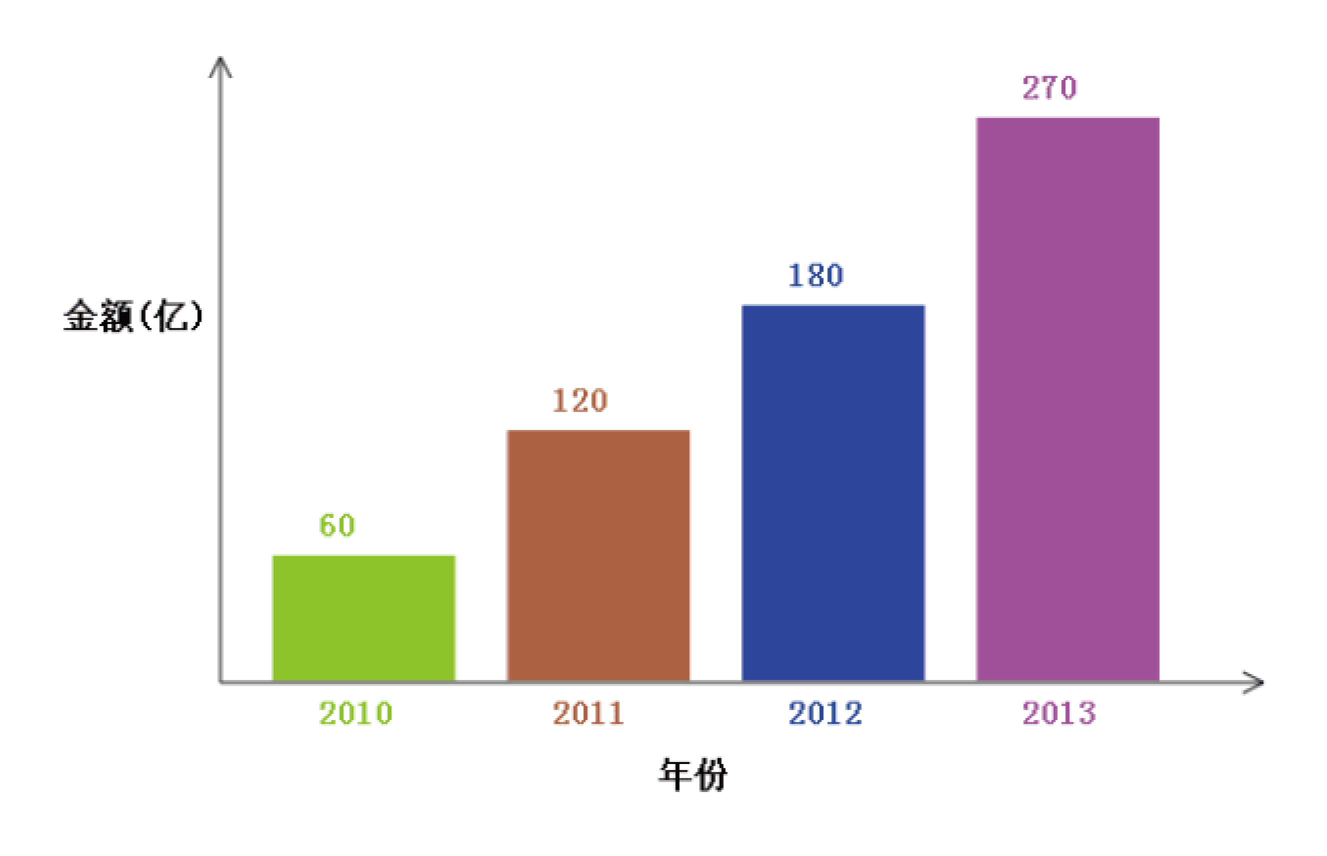


图6.26 使用Chrome打开网页文件

6.9.2 代码设计

利用编辑器打开“009.一个HTML 5版销售数据图表.html”文件，代码如下：

```
<!DOCTYPE HTML>
<html>
<body>
    <header><h2>一个HTML 5版销售数据图表</h2></header>
    <section
        <!-- 柱形图 -->
        <canvas id="chart" width="600" height="420"></canvas>
    </section>
```



本示例通过3G网络访问，需要通过Web服务器来访问网页文件。



图8.21 在iPhone 4S上用Safari浏览器打开文件

移动当前位置，行走一段距离，随着移动过程中，Google地图上会画出其移动轨迹，如图8.22所示。



可以通过开车或者坐公共交来移动当前位置，效果更佳。

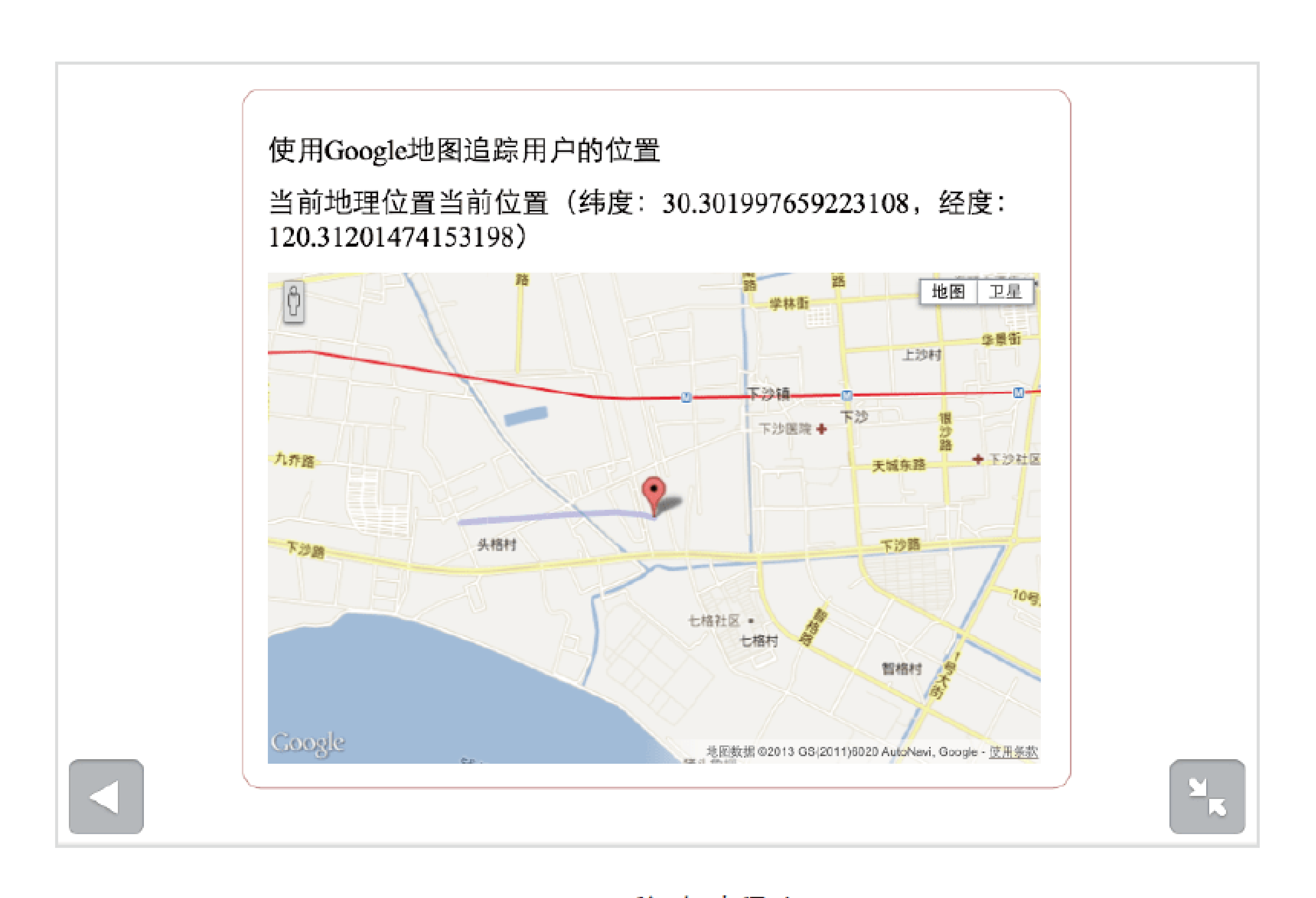


图8.22 移动过程中

8.8.2 代码设计

利用编辑器打开“用户自定义的地理定位.html”文件，代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>使用Google地图追踪用户的位置</title>
<style>
      body{
      margin:50px auto;width:634px;padding:20px;border:1px
 solid #c88e8e;
      border-radius: 15px;
        height: 100%;                         /* 设置高度自适应 */
    }
    #map{ height: 400px; width: 630px; text-align: center;}
/* 设置地图宽高 */
</style>
</head>
<body>
    <p>使用Google地图追踪用户的位置</p>
    <p>当前地理位置<span id="info"></span></p>
    <div id="map">加载中...</div>              <!-- 地图显示控件 -->
</body>
<script src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script>
;(function(){
var
gmap = document.getElementById("map"),        // 获取地图DOM
ginfo = document.getElementById("info"),      // 获取显示经纬度DOM
chinapos = new google.maps.LatLng(35.86166, 104.195397),
      // 设置默认为中国地图坐标
map = new google.maps.Map(document.getElementById("map"), {
// Google 地图实例化
    zoom: 5,
    center: chinapos,
    mapTypeId: google.maps.MapTypeId.ROADMAP
}),
marker = new google.maps.Marker({position: chinapos, map: map,
 title: "用户位置"}),        // 地图浮动提示
watchMap = function(position) {
    var
    pos = new google.maps.LatLng(position.coords.latitude,
  position.coords.longitude);    // 经纬度
      ginfo.innerHTML = "当前位置（纬度：" + position.coords.latitude
                                  + "，经度：" + position.coords.
 longitude + ")";                // 显示定位结果
      map.setCenter(pos);
      map.setZoom(14);
```

```
        marker.setPosition(pos);                   // 更新位置标记
        drawPath(position.coords);                 // 根据当前经纬度画线
},
drawPath = function(){                             // 画线函数
        var
        coordinatesPathArray = [],                 // 所监听到的所有经纬度信息
        lineOption = {                             // 画线的配置选项
                strokeColor: "#9290f8",            // 线的颜色
                strokeOpacity: 0.5,                // 线的透明度
                strokeWeight: 5                    // 线的精细
        },
        coordsPath;                                // 保存Polyline的变量
        var draw = function(coords){               // 重绘函数
                coordsPath.setMap(null);           // 清除原有的线
                        // 把新的位置信息加入到数组中
                coordinatesPathArray.push(new google.maps.LatLng(coords.latitude, coords.longitude));
                lineOption.path = coordinatesPathArray;
                // 线的path配置选项
                coordsPath = new google.maps.Polyline(lineOption);
                // 利用Google API画线
                coordsPath.setMap(map);            // 在地图上显示出线
        }
        lineOption.path = coordinatesPathArray;            // 初始化第一条线
        coordsPath = new google.maps.Polyline(lineOption);
        // 初始化Polyline并赋值给coordsPath
        return draw;
}(),
updatePosition = function(){
        var
        errorHandler = function(error){                    // 定位出错处理函数
                switch(error.code){
                        case error.PERMISSION_DENIED:      // 定位失败，没有权限
                                gmap.innerHTML = "定位被阻止，请检查您的授权或者网络协议 (" + error.message + ")";
                        break;
                        case error.POSITION_UNAVAILABLE:   // 定位失败，不可达
                                gmap.innerHTML = "定位暂时无法使用，请检查您的网络(" + error.message + ")";
                        break;
                        case error.TIMEOUT:                // 定位失败，超时
                                gmap.innerHTML = "对不起，定位超时";
                                // 超时了
                        break;
                }
        },
        getWatchPosition = function(){                     // 定位函数
            var watchId = navigator.geolocation.watchPosition(watchMap, errorHandler, {timeout: 1000});
```

```
81         };
82         return getWatchPosition;                    // 返回定位函数供外部调用
83 }();
84 if (navigator.geolocation) {
85         gmap.innerHTML = "定位中...";
86         updatePosition();                                  // 定位开始
87 } else {
88         gmap.innerHTML = '您的浏览器不支持地理位置 ~o(∩_∩)o~';
           // 定位失败，浏览器不支持
89 }
90 }());
91 </script>
92 </html>
```

8.8.3 代码分析

第40行，作用是在用户移动过程中，重新标记用户的当前位置。

第41行，调用画线程序。Google提供的画线API Polyline会在两点间画出一条直线。根据用户频繁的移动位置，形成多个点，把每个点连接，就成了一条直线。

第43行~第63行，画线函数。首先把画线的变量保存在闭包中，注意第62行代码，该函数会在页面载入时立即执行，并初始化coordinatesPathArray、lineOption、coordsPath这三个变量，第61行返回draw函数。

第52行~第59行，定义真正的画线函数draw，第40行调用的drawPath函数实际上调用的是drawp函数。传递的参数为当前的坐标。draw函数的原理是先清除原先已经画的轨迹，再把当前的位置坐标加入到历史坐标数组coordinatesPathArray中，最后根据新的数组，在Google Map上重新绘制一条轨迹。

第53行，清除当前地图上已经画好的路径轨迹。

第55行~第57行，把当前新的位置坐标数组在Google Map上重新绘制出来。



更多关于通过Google Map画线的知识点，请参考Google官方网站https://developers.google.com/maps/documentation/javascript/reference#Polyline。

代码第86行是本示例的入口函数，该函数调用getWatchPosition方法，然后执行navigation.geolocation对象的watchPosition方法。

第80行，调用HTML 5 Geolocation API的watchPosition函数，其有三个参数，这三个参数和getCurrentPostion含意一样，区别在于watchPosition函数是一个监视器，监视用户的位置是否发生变化，如果发生变化，浏览器就会触发其回调函数，成功则回调函数watchMap，失败则回调函数errorHandler。



8.9 示例9 使用Google地图查找路线

8.9.1 示例效果

本示例演示一个生活中经常用到的场景，根据Google地图查找出行路线。路线查找需要提供起始位置和目的地位置。利用HTML 5提供的获取地理位置信息，可以非常方便地定位到当前地理位置，然后提供目的地，就可以根据Google地图API查找出行路线。本示例演示的路线查找功能也可以选择出行方式，包括自驾车、公交、步行、自行车这4种方式。

使用Chrome浏览器打开“使用Google地图查找路线”网页文件，运行效果如图8.23所示。



图8.23 查找路线页面

在起始位置一栏单击“使用当前位置作为起始位置”文字按钮，运行效果如图8.24所示。

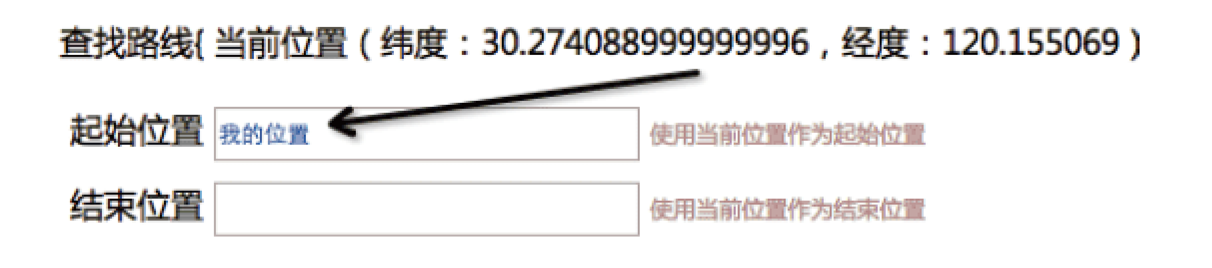


图8.24 单击使用当前位置作为起始位置

在结束位置一栏，填写“西溪国家湿地公园”，出行方案选择“公交”，然后单击查找按钮，运行效果如图8.25所示。图中左侧以地图形式显示查找结果，标签A代码起始位置，标签B代表结束位置，图中右侧以文字的形式显示具体的路线信息。



图8.25 查找去西溪国家湿地公园的路线

在出行方案选项卡中，单击下拉框选择驾车，然后单击“查找”按钮，运行效果如图8.26所示。



图8.26 驾车路线

在出行方案选项卡中，单击下拉框选择自行车，然后单击“查找”按钮，运行效果如图8.27所示。

8.9.2 代码设计与分析

利用编辑器打开“使用Google地图查找路线.html”文件，样式部分代码如下：

```
<style>
   body{
    margin:50px auto;width:870px;padding:20px;border:1px solid #c88e8e;
    border-radius: 15px;
        height: 100%;                              /* 设置高度自适应 */
    }
    .item { width:430px; display: inline-block;padding-right:2px;}
   /* 设置ip和wifi容器的宽度并左浮动 */
    .section{padding: 5px;}
    .btn{text-decoration: none; color: #c89191;font-size: 11px; }
    .btn:hover{text-decoration: underline;}
    input, select{border: #b9aaaa 1px solid; height: 22px;width:
    200px;margin-left:5px;}
    #map{ height: 400px; width: 430px; text-align: center;}
   /* 设置地图宽高 */
    .search{                                       /* 设置查找按钮样式 */
         padding: 4px 12px;
         text-decoration: none;
         cursor: pointer;                          /* 设置光标的手形 */
         color: #333333;
         background-color: #f5f5f5;                /* 设置查找按钮背景色 */
         border-radius: 4px;
         box-shadow: inset 0 1px 0 rgba(255,255,255,.2),
 0 1px 2px rgba(0,0,0,.05);/* 设置查找按钮阴影 */
    }
</style>
```

HTML部分代码如下：

```
<p>查找路线<span id="info"></span></p>
    <div class="section">                          <!--起始位置 -->
      <label for="start">起始位置</label><input type="text" id=
"origin" />                    <!--起始位置输入框—>
      <a href="javascript:;" class="btn" id="user-origin">使用当前位置作
 为起始位置</a>
    </div>
    <div class="section">                          <!--结束位置 -->
          <label for="end">结束位置</label><input type="text"
 id="destination" />           <!--结束位置输入框 -->
          <a href="javascript:;" class="btn" id="user-destination">
 使用当前位置作为结束位置</a>
    </div>
    <div class="section">
          <label for="travelMode">出行方案</label>     <!--出行方案选择-->
          <select id="travelMode">
                 <option value="TRANSIT">公交</option><!--选择公交-->
                 <option value="DRIVING">驾车</option><!--选择驾车-->
```

```
        map = new google.maps.Map(gmap, options),     // 地图
        // 地图位置标记
        currentMaker = new google.maps.Marker({position: currentPosition, map: map, title: "用户位置"});
        ginfo.innerHTML = "{ 当前位置（纬度：" + position.coords.latitude
                                  + "，经度：" + position.coords.longitude + "）}";// 显示定位结果
        directionsDisplay.setMap(map);                      // 在地图上显示路线
        directionsDisplay.setPanel(directionsPanel);    // 显示路线查找文字结果
},
userSelectionCurrent = function(e){             // 设置当前位置作为查找点
        var prev = this.previousElementSibling;          // 获取input元素
        prev.value = '我的位置';                          // 设置input元素的值
        prev.style.color = 'blue';                  // 将input元素字体设置为蓝色
        prev.isCurrent = true;              // 设置input使用当前位置来计算
},
cancelCurrent = function(){
        this.style.color = '#111';          // 设置input元素字体颜色为#111
        this.isCurrent = false;             // 设置不使用当前位置作为查找点
},
bind = function(){
        [userOrigin, userDestination].forEach( function (item){
                item.addEventListener('click', userSelectionCurrent, false);    // 绑定使用当前位置的单击事件
        // 如果input元素的值为人为改变，则设置不使用当前位置作为查找点
        item.previousElementSibling.addEventListener('change', cancelCurrent, false);
        });
        search.addEventListener("click", calcRoute, false);     // 绑定查找按钮事件
},
calcRoute = function(){                         // 路线查找函数
        var
        start = origin.isCurrent ? currentPosition : origin.value,      // 获取路线的起始位置
        end = destination.isCurrent ? currentPosition : destination.value,        // 获取路线的结束位置
        selectedMode = travelMode.value,        // 获取路线的出行方式
        request = {                             // 封装route函数参数
                origin:start,
                destination:end,
                travelMode: google.maps.TravelMode[selectedMode]
        };
        // 调用Google地图API请求路线
        directionsService.route(request, function(response, status) {
                if (status == google.maps.DirectionsStatus.OK) {     // 路线找到
                        directionsPanel.innerHTML = '';   // 清除文字结果
                        directionsPanel.style.color = '';// 清除文字结果颜色
                        directionsDisplay.setMap(map);    // 在地图上显示路线
```

```
                    directionsDisplay.setDirections(response);
                    // 显示文字结果
                    currentMaker.setMap(null);       // 清除位置标记
                }else{
                    directionsPanel.style.color = 'red';
                    // 没有找到路线
                    if(status === google.maps.DirectionsStatus.ZERO_RESULTS){
                        // 自行车查找的特殊处理
                        if(selectedMode === 'BICYCLING'){
                            directionsPanel.innerHTML = '没有找到路线，可能是不支持当前国家';
                        }else{
                            directionsPanel.innerHTML = '没有找到相关路线';
                        }
                    }else if(status == google.maps.DirectionsStatus.NOT_FOUND){
                        directionsPanel.innerHTML = '地址没有找到';
                    }else{
                        directionsPanel.innerHTML = '其他错误：' + status;
                    }
                    directionsDisplay.setMap(null);// 清除上一次显示的路线
                    currentMaker.setMap(map);        // 显示当前的位置标记
                }
        });
},
getPosition = function(){
        var
        errorHandler = function(error){          // 定位出错处理函数
                switch(error.code){
                    case error.PERMISSION_DENIED:     // 定位失败，没有权限
                        gmap.innerHTML = "定位被阻止，请检查您的授权或者网络协议（" + error.message + "）";
                    break;
                    case error.POSITION_UNAVAILABLE:  // 定位失败，不可达
                        gmap.innerHTML = "定位暂时无法使用，请检查您的网络（" + error.message + "）";
                    break;
                    case error.TIMEOUT:               // 定位失败，超时
                        gmap.innerHTML = "您的网络较慢，请耐心等待...";
                        gmap.innerHTML = "对不起，定位超时";
                        // 超时了
                    break;
                }
        },
        getCurrentPosition = function(){             // 定位函数
            navigator.geolocation.getCurrentPosition(showMap, errorHandler);
        };
```

```
103        return getCurrentPosition;                    // 返回定位函数供外部调用
104 }();
105 var init = function(){
106        if (navigator.geolocation) {
107               gmap.innerHTML = "定位中...";
108               getPosition();                         // 定位开始
109               bind();
110        } else {
111               gmap.innerHTML = '您的浏览器不支持地理位置 ~o(∩_∩)o~';
               // 定位失败，浏览器不支持
112        }
113 };
114 google.maps.event.addDomListener(window, 'load', init);
    // 入口函数
```

第17行，在navigator.geolocation.getCurrentPosition函数的回调结果中，用currentPosition记录定位的结果。如果是使用当前位置进行查找路线，这个结果在执行路线查找时会用到。

第28行~第33行，定义“使用当前位置作为起始位置”和“使用当前位置作为结束位置”的事件处理函数，当用户单击其按钮时，设置其对应的文本输入框的值为“我的位置”，并把字体颜色改为蓝色。然后设置变量isCurent的值为true，用来标记要使用当前位置作为起始或者结束位置。

第34行~第37行，取消使用当前位置为作查找条件。



当用户在输入框中敲入文字时，表示用户不想使用当前位置来查找。

第38行~第45行，定义了“使用当前位置作为起始位置”、“使用当前位置作为结束位置”、两个文本输入框和“查找”按钮的事件代理。

第46行~第83行，是查找路线的处理函数。

第48行~第49行，获取路线查找的起始和结束位置，如果使用当前位置，则其值为第17行代码赋予变量currentPostion的值，如果不使用当前位置作为查找条件，则对应获取用户输入的文字。

第50行，获取用户选择的出行方式。

第57行，调用directionsService的route方法，该方法提供两个参数。第一个参数为查找条件（包括路线的起始、结束位置和出行方式等），第二个参数为查找结果的回调函数。回调函数中第一个参数是具体的路线结果，第二个参数代表查找结果的状态。



了解更多谷歌地图相关的查找路线的信息接口，读者可以参考https://developers.google.com/maps/documentation/javascript/reference#DirectionsService。

第58行，表示已经查找到了结果。

第61行，如果查找到路线，在地图上显示出路线。

第62行，在<div id="directionsPanel">上显示查找结果的文字方案。

第65行~第78行，是没有正确查找到路线的错误处理逻辑，常见的有三种错误。



第 9 章 拖放大演练

拖曳在HTML 5出现之前是Web 2.0时代非常流行的技术，市面上基本所有的JavaScript类库都有拖曳模块。在HTML 5出现之后，任何DOM元素都可以被拖曳，只需要在DOM元素上简单地设置属性draggable为true，然后就可以调用脚本来监听拖曳的各种事件，如拖曳开始、拖曳中、拖曳结束、进入释放位置区域等。另外，拖曳功能还能和HTML 5的File API功能相互结合，实现拖曳文件上传目的，这些功能的使用都会出现在本章的示例之中。

本章知识点：

- 网页中各个元素的拖放
- 制作一个可拖放的阅读器



9.1 示例1 实现网页元素的拖放

9.1.1 示例效果

拖放事件是HTML 5标准的一部分，本实例将结合此标准模拟桌面应用丢弃废文件的动画过程。页面上有3个纸团，将纸团拖向蓝色纸篓，纸篓会由空置转为填满状态，并发出投入纸团的声音。使用Chrome浏览器打开网页文件，运行结果如图9.1所示。

单击左侧1号纸团，拖动到纸篓上方，此时光标状态由“拖动”变为“放置”。松开鼠标，此时1号纸团消失，纸篓左右摇摆，同时纸篓状态由空置变为被填满状态，音频发出纸团被丢入纸篓的声音，纸篓前的数字由0变为1，运行效果如图9.2所示。

依次拖入2号、3号纸团，重复出现上述过程。



图9.1 使用Chrome打开网页文件



图9.2 拖动1号纸团并丢入纸篓



本例代码在Chrome下测试通过。

9.1.2 代码设计

利用编辑器打开“实现网页元素的拖放.html”文件，代码如下：

```
<!doctype html>
<html>
<head>
    <style>
    .garbage{ width:120px; height:135px; position: relative; }
/* 纸篓样式类 */
    .garbage i{                                         /* 纸篓纸团计数 */
       position: absolute; top: 61px; left: 46px;
       font-weight:bold; color:red;
    }
    .empty{ background:url(../images/garbage-e.png) no-repeat; }
 /* 空状态纸篓 */
    .full{ background:url(../images/garbage-f.png) no-repeat; }
/* 满状态纸篓 */
    .paper{                                             /* 纸团样式类 */
        background:url(../images/paper.png) no-repeat;float:left;
        width:64px; height:57px; cursor:move; position: relative;
    }
    .paper i{                                           /* 纸团编号 */
       position: absolute; top: 18px;  left: 23px;
       font-weight:bold; color:black;
    }
    .line_3d{                                    /* 垃圾桶摇摆样式类 */
        -webkit-animation-name:shake;
        -webkit-animation-duration:1s;
    }
    @-webkit-keyframes shake{                    /* 定义投放垃圾动画名 */
        from{-webkit-transform:rotate(0deg);}
        20%{-webkit-transform:rotate(20deg);}
        60%{-webkit-transform:rotate(-20deg);}
        to{-webkit-transform:rotate(0deg);}
    }
    </style>
</head>
<body>
    <header><h2>实现网页元素的拖放</h2></header>
    <section>
        <!-- 垃圾桶 -->
        <div class="garbage empty"><i>0</i></div>
        <!-- 垃圾丢放音频 -->
        <audio src="../res/garbage.mp3">您的浏览器不支持audio标签</audio>
    </section>
```

代码第38行为音频元素，用于播放纸团投入纸篓时的声音。

代码第42行~第44行为三个纸团的HTML结构，每个元素都设置draggable属性为true，表明该元素允许被拖动。

代码56行~第60行，循环三个纸团元素，分别监听元素的dragstart事件，当元素开始被拖动时触发该事件。事件触发后，将目标纸团元素赋值于闭包中的current_paper变量。

代码61行~第63行，监听纸篓元素的dragover事件，并在被触发时，调用方法preventDefault，阻止浏览器的默认行为，否则无法触发纸篓元素drop事件。

代码64行~第75行，完成整个动画的后半程效果。当纸团投入纸篓，首先，调用音频元素的play方法播放声音。接着，给纸篓的样式加入摇摆动画样式类“line_3d”，并设置一个计时器，在1s动画结束后移除纸篓的动画样式类“line_3d”。

代码第70行~第72行，通过正则表达式判断纸篓样式类中是否含有值为full样式类（表示纸篓填满状态），如果存在则不继续添加。

在代码第58行，将当前触发的拖动纸团元素保存在current_paper变量中，代码第74行利用该保存的变量，将当前拖入纸篓的纸团元素从DOM中移除。



9.2 示例2 拖放图标

9.2.1 示例效果

本例使用HTML 5模拟移动设备拖放功能。整个示例外观很像一个平板电脑界面，页面摆放着各种应用图标，使用Chrome浏览器打开网页文件，运行结果如图9.3所示。

单击其中一个图标，并在屏幕上进行拖动，运行结果如图9.4所示。



图9.3 使用Chrome打开网页文件



图9.4 鼠标拖动图标



读者可以尝试多次拖动图标，图标无法超越矩形边界。

9.2.2 代码设计

利用编辑器打开“拖放图标.html”文件，代码如下：

```
<!doctype html>
<html>
<head>
    <style>
    .phone{                                          /* 模拟平板外宽 */
        background:url(../images/phone.jpg) no-repeat;
        background-position:-287px -84px;
        position: relative; width: 450px; height: 550px;
        -moz-border-radius: 10px;                    /* 屏幕圆角效果 */
        -webkit-border-radius: 10px;
        border-radius: 10px;
    }
    img{ width:59px; height:60px; cursor:pointer; position: absolute;}
    .facebook{ top: 48px; left: 45px; }       /* facebook图标位置 */
    .contacts{ top: 48px; left: 145px;}              /* 通讯录图标位置 */
    .calculator{ top: 48px; left: 245px;}            /* 计算器图标位置 */
    .camera{ top: 48px; left: 345px;}                /* 照相机图标位置 */
    .pro{top: 165px; left: 45px;}                    /* 游戏图标位置 */
    .flashlight{top: 165px; left: 145px;}            /* 手电图标位置 */
    .copiercin{top: 165px; left: 245px;}             /* 文章图标位置 */
    .bossprefs{top: 165px; left: 345px;}             /* 开关图标位置 */
    </style>
</head>
<body>
    <header><h2>拖放图标</h2></header>
    <section>
        <div class="phone">
            <!-- 图标按钮列表 开始 -->
            <img class="facebook" src="../images/icon/facebook.png" />
            <img class="contacts" src="../images/icon/contacts.png" />
            <img class="calculator" src="../images/icon/calculator.png" />
            <img class="camera" src="../images/icon/camera.png" />
            <img class="pro" src="../images/icon/pro.png" />
            <img class="flashlight" src="../images/icon/flashlight.png" />
            <img class="copiercin" src="../images/icon/copiercin.png" />
            <img class="bossprefs" src="../images/icon/bossprefs.png" />
            <!-- 图标按钮列表 结束 -->
        </div>
    </section>
</body>
<script>
    (function () {
        var slice = Array.prototype.slice,
            phone = document.querySelector('div.phone'),   // 圆角屏幕
            icons = slice.call(document.querySelectorAll('img'), 0);
             // 所有图标的元素数组
```

元素的页面坐标不能超出浏览器之外，同时，获取矩形苹果界面背景左上、左下、右上、右下4个点的坐标，判断图标的元素坐标在此区域之内。



9.3 示例3 设置拖放的效果

9.3.1 示例效果

HTML 5不仅提供了更强的元素拖放功能，同时还在拖放效果增强上给出了更多选择。本例延续丢废纸的场景，在拖放的同时加入图片跟随和图标变换效果。使用最新版本Opera浏览器打开网页文件，运行结果如图9.5所示。



经测试，最新版本Chrome浏览器在支持setDragImage等关键API上存在问题，本例建议采用最新版本Opera打开运行。

单击1号纸团，并拖至垃圾桶，效果如图9.6所示。松开鼠标，1号纸团消失，垃圾桶被填满。依次拖入2号、3号纸团，效果与1号纸团相同。



图9.5 使用Opera打开网页文件



图9.6 将1号纸团拖至垃圾桶

9.3.2 代码分析

本例代码的主体逻辑部分与“实现网页元素的拖放.html”相同，下面就增强部分做一个分析。

监听纸团元素的两个拖放事件dragstart和dragend，代码如下：

```
paper.addEventListener('dragstart', function (e) {   // 监听纸团拖动开始事件
    current_paper = paper;                  // 设置被拖动的纸团于闭包变量中
    e.dataTransfer.effectAllowed = 'move';            // 允许拖动操作类型
    e.dataTransfer.setDragImage(current_paper, 10, 0);// 设置拖放时效果
    paper.style.opacity = '0.4';                      // 设置纸团透明度
}, false);
paper.addEventListener('dragend', function (e) {   // 监听纸团拖放结束事件
```

```
08      paper.style.opacity = '1';                      // 恢复纸团的透明度
09  }, false);
```

代码第03行，设置e.dataTransfer的effectAllowed属性值为move，表示允许拖放元素执行move行为。除了示例中的move外，还有其他几种行为，说明如下。

- uninitialized：浏览器默认拖放行为，表示未被设置。
- none：被拖动的元素不能有任何行为。
- copy：只允许值为copy的dropEffect。
- link：只允许值为link的dropEffect。
- copyLink：允许值为copy和link的dropEffect。
- copyMove：允许值为copy和link的dropEffect。
- linkMove：允许职位link和move的dropEffect。
- all：允许任意dropEffect。

其中，如果effectAllowed值为uninitialized时，又分为三种情况，说明如下：

- 拖放元素为文本框中选中的内容：允许值为move、copy和link的dropEffect。
- 拖放元素为普通选中项：允许值为copy、link和move的dropEffect。
- 拖放元素为带链接的a元素：允许值为link、copy、move的dropEffect。

代码第04行是实现拖放效果的关键，语法如下：

```
datatransfer.setDragImage(element, x, y)
// 使用元素替换拖曳反馈，其中x和y表示相对于鼠标的坐标
```

上面讲到dropEffect属性，读者可能有些不明白，下面看看如何使用这个属性，代码如下：

```
01  garbage.addEventListener('dragover', function (e) {
    // 鼠标拖曳在该元素上移动时触发
02      e.preventDefault();                         // 阻止浏览器元素默认时间
03      e.dataTransfer.dropEffect = 'move';         // 拖放移动中的效果
04  }, false);
```

注意，在第03行代码中，dropEffect属性被设置为与当初effectAllowed相同的值move，表示目标元素drop事件允许被监听move行为，dropEffect属性有4种可能的值，说明如下。

- none：不能被拖放入元素。
- move：允许把拖放元素移动到放置目标。
- copy：允许把拖放元素复制到放置目标。
- link：拖放元素必须是a元素，放置目标会打开被拖放元素。



还有更多拖放功能本例没有充分展现，想了解更多关于拖放效果的调用方法和功能，可以参考网站https://developer.mozilla.org/en-US/docs/DragDrop/DataTransfer。

```
            font-family: 'Gochi Hand', cursive;
            font-size: 25px; width: 320px;
            border: 2px solid black; padding:5px;
            border-radius: 10px;                            /* 圆角 */
            -moz-border-radius: 10px;  -webkit-border-radius: 10px;
            background-color: #f1f1f1;
        }
        img {width:70px; height:70px; margin:0 5px;}     /* 商品图片样式 */
        div{                                             /* 单个商品样式 */
            border-bottom: 2px solid black;
            cursor: move; padding-top:5px;
            -webkit-transition: -webkit-transform 0.2s ease-out;   /* CSS 3动画 */
            -moz-transition: -moz-transform 0.2s ease-out;
            transition: transform 0.2s ease-out;
        }
        .over {background-color: #FFFFD5;}        /* 拖动悬浮元素样式 */
        .moving {                                 /* 被拖动元素样式 */
            opacity: 0.25;
            -webkit-transform: scale(0.97);       /* 元素缩小 */
            -moz-transform: scale(0.97);  transform: scale(0.97);
         }
        </style>
    </head>
    <body>
        <header><h2>对照片进行排序</h2></header>
        <section>
        <!-- 商品元素列表 -->
        <div draggable="true"><img src="../images/shop/黑色护腕.png" draggable="false"/>黑色护腕</div>
        <!-- ......省略其他商品结构，可以查看本书网络资源-->
        </section>
    </body>
    <script>
        (function () {
            var slice = Array.prototype.slice,
                list = slice.call(document.getElementsByTagName('div'), 0),  // 商品元素列表数组
                move_item;                            // 当前拖动元素
            list.forEach(function (item) {            // 循环数组，监听元素事件
                item.addEventListener('dragstart', function (e) {  // 拖动开始触发
                    e.dataTransfer.effectAllowed = 'move';  // 设置拖动允许的事件行为
                    e.dataTransfer.setData('html', e.target.innerHTML);  // 保存目标元素结构至拖曳数据中
                    this.style.opacity = '0.4';
                    this.classList.add('moving');
                    move_item = item;                 // 将目标元素保存在闭包变量中
                }, false);
                item.addEventListener('dragenter', function (e) {
```

```
                  // 拖入元素边界内触发
55                    e.target.classList.add('over');
56                }, false);
57                item.addEventListener('dragover', function (e) {
                // 元素在目标元素上移动
58                    e.preventDefault();     // 阻止默认事件，保证drop被触发
59                    e.target.classList.add('over');
60                }, false);
61                item.addEventListener('dragleave', function (e) {
                // 元素离开目标元素触发
62                    e.target.classList.remove('over');
63                }, false);
64                item.addEventListener('drop', function (e) {
                // 拖曳元素放在目标元素内
65                    var target = e.target;
66                    if (move_item != target) {     // 拖动元素与放置元素不同
67                        move_item.innerHTML = item.innerHTML;
                        // 将被拖放元素结构填充为放置元素
68                        item.innerHTML = e.dataTransfer.getData('html');
                        // 设置拖放元素结构为被拖放元素
69                    };
70                }, false);
71                item.addEventListener('dragend', function (e) {
                // 拖曳完毕后触发
72                    this.style.opacity = '1';
73                    list.forEach(function (item) {
74                        item.classList.remove('moving');     // 移除拖动样式
75                    });
76                }, false);
77            });
78        })();
79 </script>
80 </html>
```

9.4.3 代码分析

代码从46行开始，监听商品列表元素的拖曳相关事件。首先看dragstart事件，设置dataTransfer的effectAllowed属性为move，允许拖曳的移动事件行为。调用dataTransfer的setData方法，存储目标元素HTML结构，该方法语法如下：

```
event.dataTransfer.setData(dataFormat,data);
// 存放被拖数据的值，其中dataFormat为字符类型
```

接着设置目标元素的透明度为0.4，表示该元素正在被执行拖曳操作，并为该元素添加moving样式类。moving样式类带有一个CSS 3的动画效果，将元素的尺长缩减至原来的0.97倍。在dragstart的事件最后，将目标元素缓存至闭包中的move_item变量内，后面会在drop事件中进行使用。

触发dragenter和dragover这两个事件后，要做的事情很相似，给被进入和被悬浮的元素添加over样式类，该样式类会改变元素的背景色为淡黄色。值得注意的是，在dragover监听事件

中，会执行事件对象的preventDefault方法，阻止元素默认行为，确保元素的drop事件被正确触发。

拖曳元素离开商品列表时，触发dragleave事件，此时移除被进入元素的over样式类，取消背景淡黄色的效果。

当拖曳的元素被放置在商品列表其中的任意元素，触发drop事件。触发后，首先判断被拖曳元素和放置的元素是否为同一元素，如果为同一元素，表明元素被放置于自身之上，此时不执行任何动作。如果不是，则设置被拖曳元素的HTML结构为目标元素的HTML结构。同时，将目标元素的结构设置为被拖曳元素结构，这里用到了dataTransfer的getData方法，获取在dragstart事件触发时存入的被拖曳元素的HTML结构字符，该方法语法如下：

```
event.dataTransfer.getData(dataFormat);    // 从指定字符中获取存放被拖数据的值
```

dragend事件会在所有拖曳事件结束后被触发。触发后，恢复目标元素的透明度，并循环移除所有元素的moving样式类。



示例中绑定元素的事件顺序与元素被拖曳后触发的事件顺序相同，按事件触发的先后顺序排序为：dragstart、dragenter、dragover、dragleave、drop、dragend。

9.4.4 相关知识

1. @font-face

在CSS 3出现之前，设计师和开发人员只能使用计算机上安装好的字体。CSS 3带来了“@font-face”规则，允许使用任何字体，可以将字体存放在服务器上，需要呈现时被下载到用户计算机。本例中的使用如下：

```
@font-face {
    font-family: 'Gochi Hand';
    src:local('GochiHand'),local('GochiHand-Regular'), url('http://******.
   woff') format('woff');
}
section{
    font-family: 'Gochi Hand', cursive;
}
```

- font-family：规定了字体的名称。
- src：定义字体文件的URL。字体类型有woff文件后缀的格式，还有ttf和eot等格式。
- local：表示本地的字体文件名。

虽然网络字体的好处不言而喻，但是中文网络字体由于其体积庞大加上版权问题，所以离实际的使用还有一段距离。

```
var slice = Array.prototype.slice,
    list = slice.call(document.getElementsByTagName('div'), 0),
    // 商品元素列表数组
    section = document.querySelector('section'),
    move_item,                                      // 当前拖动元素
    reader,
    events = {
        'dragstart': function (e) {                 // 拖动开始触发
            // ......省略(与示例“对照片进行排序”基本相同),请参看本书网络资源
        },
        'dragenter': function (e) {                 // 拖入元素边界内触发
            var files = e.dataTransfer.files;
            if (files && !files.length) {// 判断为非拖入文件情况
                e.target.classList.add('over');
            };
        },
        'dragover': function (e) {                  // 元素在目标元素上移动
            // ......省略(与示例“对照片进行排序”基本相同),请参看本书网络资源
        },
        'dragleave': function (e) {                 // 元素离开目标元素触发
            // ......省略(与示例“对照片进行排序”基本相同),请参看本书网络资源
        },
        'drop': function (e) {                      // 拖曳元素放在目标元素内
            // ......省略(与示例“对照片进行排序”基本相同),请参看本书网络资源
        },
        'dragend': function (e) {                   // 拖曳完毕后触发
            // ......省略(与示例“对照片进行排序”基本相同),请参看本书网络资源
        }
    };
list.forEach(function (item) {                      // 循环数组,监听元素事件
    for (var key in events) {
        item.addEventListener(key, events[key], false);
        // 监听元素事件
    };
});
section.addEventListener('drop', function (e) {
// 监听列表容器的drop事件
    e.preventDefault();
    var files = e.dataTransfer.files;      // 获取从系统拖入的文件对象列表
    if (!files || !files.length) {         // 没有文件对象拖入情况
        return;
    };
    files = slice.call(files, 0);          // 将文件列表转为数组
    files.forEach(function (file) {
        if (file.type.toLowerCase().match(/image.*/)) {
        // 判断是否为图片文件
            reader = new FileReader(); // 实例化FileReader对象,读取文件数据
            reader.addEventListener('load', function (e) {
            // 监听FileReader实例的load事件
               var item = document.createElement('div');
               item.setAttribute('draggable', true);
             // 设置元素允许拖动
```

```
47                  item.innerHTML = '<img src="' + e.target.result +
   '" draggable="false"/>' + file.name.split('.')[0];
48                   for (var key in events) {      // 循环绑定新建元素事件
49                       item.addEventListener(key, events[key], false);
50                   };
51                   section.appendChild(item);     // 将商品元素添加到列表中
52               });
53               reader.readAsDataURL(file);        // 读取文件为DataURL
54           };
55       });
56 });
57 section.addEventListener('dragleave', function (e) {
   // 鼠标拖曳离开该元素时触发
58     e.preventDefault();
59 }, false);
```

9.5.3 代码分析

代码06行~第28行，定义了一个事件对象字面量events，里面包含了事件dragstart、dragenter、dragover、dragleave、drop、dragend。其中要注意的是代码第12行，该条判断保证事件触发由页面中的商品列表元素产生，避免由本地文件夹文件拖动造成。

代码第29行~第33行，循环页面上的商品元素列表，分别绑定events对象字面量内的各个事件。



由于后续新建的商品元素还需要绑定该事件列表，所以本例将各种事件对象独立到一个对象变量中。

先看最后三行代码，监听商品列表容器section元素的dragleave事件。触发后调用事件对象的preventDefault方法，阻止元素的默认行为，确保drop事件被正确调用。

代码第34行~第56行，主要完成了在文件放置于列表后，将图片和图片文件名添加到商品列表，并绑定排序相关事件的这一系列动作。从代码第41行开始，循环由drop事件获取的文件列表，判断文件类型是否为图片类型。当遇到图片类型文件，新建FileReader对象，读取文件内容。在读取完毕的load事件回调函数中，新建一个DIV商品元素容器，填充图片和文字，并循环绑定events事件，最后添加到商品列表中。添加完毕后，商品列表的每个元素之间可以相互拖曳，效果与示例“对照片进行排序”相同。



9.6 示例6　将商品拖入购物车

9.6.1 示例效果

经常上电子商务网站的朋友都非常喜欢购物车这个功能。传统的购物车，一般在每个商品周围会有一个“添加购物车”按钮，用户单击该按钮，购物车中就会出现对应的商品。本

例将结合HTML 5拖曳功能，实现更加贴近现实的添加购物车功能，用户可以使用鼠标将商品直接拖曳入购物车。

使用Chrome浏览器打开网页文件，结果如图9.12所示。

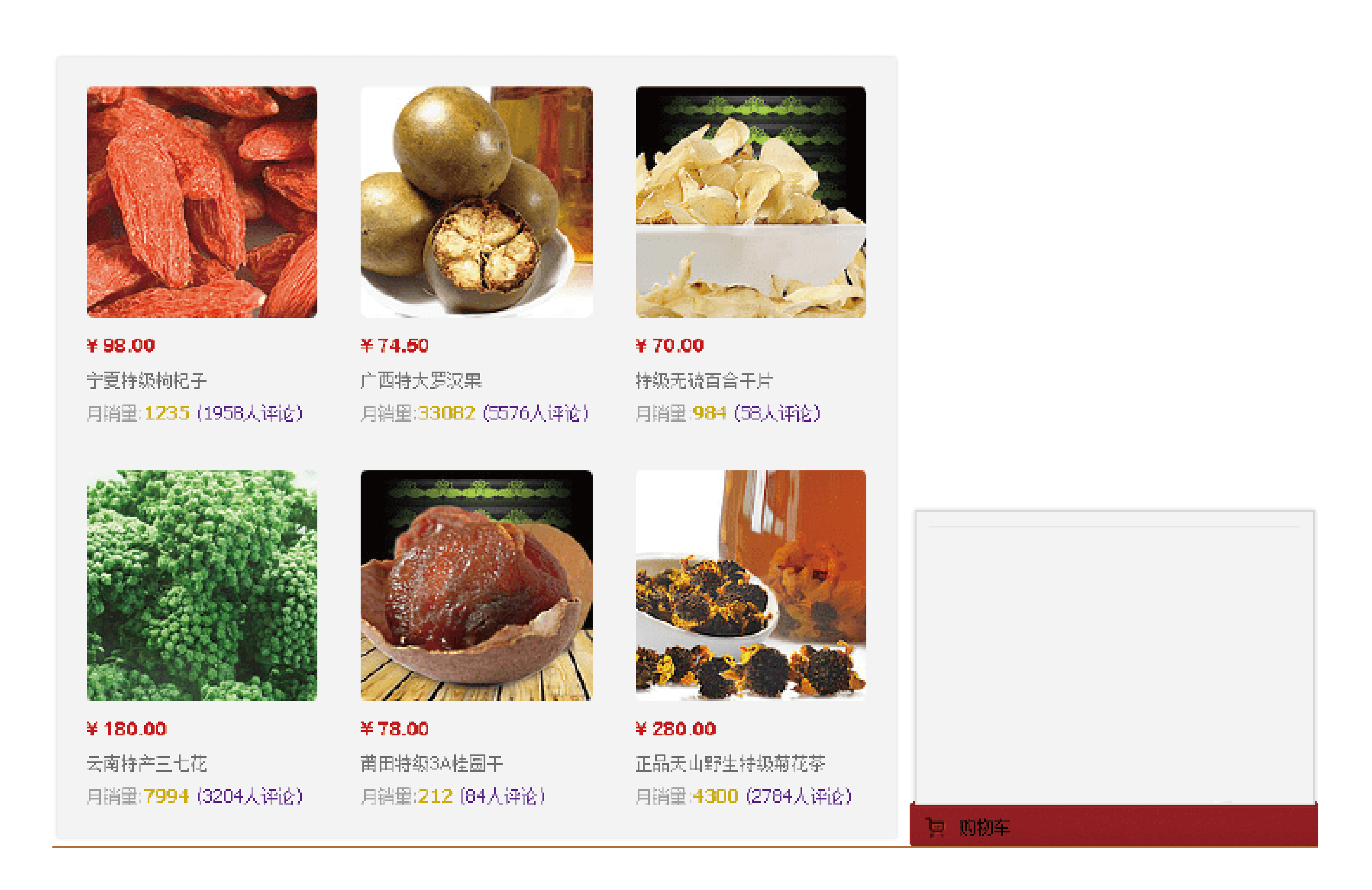


图9.12 使用Chrome打开网页文件

单击左侧商品列表的第一件商品，并拖曳入右下角购物车内，被拖曳商品尺寸变小并呈现透明，效果如图9.13所示。

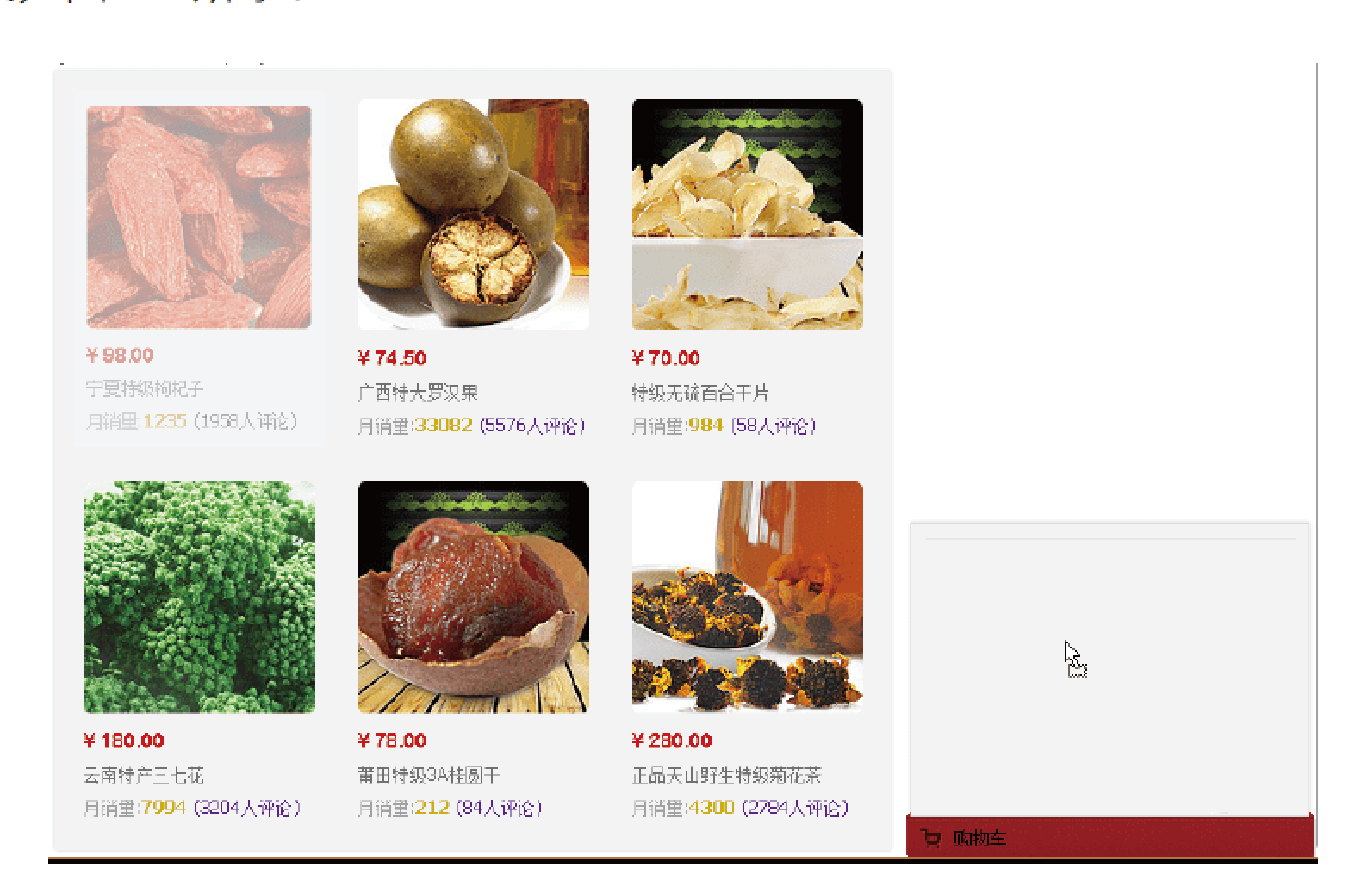


图9.13 拖曳第一件商品入购物车

在购物车上方松开鼠标，此时，第一件商品在商品列表中消失，并慢慢的从购物车中出现，效果如图9.14所示。

```
60 });
61 list_items.bind({                                    // 绑定商品列表若干事件
62     'mouseenter': function () { $(this).addClass('product-hover'); },
       // 鼠标移入效果增强
63     'mouseleave': function () { $(this).removeClass('product-hover'); },
       // 移出效果消失
64     'dragstart': function (e) {
65         var self = $(this);
66         e.originalEvent.dataTransfer.effectAllowed = 'move';
           // 设置拖动允许的事件行为
67         self.css('opacity', 0.4).addClass('moving');
           // 修改元素样式并添加动画
68         current_item = self;                         // 记录当前的拖曳元素
69     },
70     'dragleave': function (e) { e.preventDefault(); },
       // 阻止默认事件触发
71     'dragend': function (e) { $(this).css('opacity',
 1).removeClass('moving') }       // 拖曳结束元素样式恢复
72 });
73 ghandle.bind({                                       // 绑定购物车若干事件
74     'dragover': function (e) {
75         e.preventDefault();                          // 阻止浏览器元素默认时间
76         e.originalEvent.dataTransfer.dropEffect = 'move';
           // 设置拖放移动中的效果
77     },
78     'drop': function (e) {                           // 商品元素放置购物车
79         var item = $(substitute(tpl, {               // 获取商品元素信息
80             path: current_item.find('img').attr('src'),
               // 图片地址
81             title: current_item.find('h3.product-title a').text(),
               // 商品描述
82             price: current_item.find('span.ui-price strong').text()
               // 商品价格
83         }));
84         item.data('product', current_item).css('opacity', 0);
           // 缓存商品并设置透明
85         current_item.animate({ opacity: 0 }, 500, function ()
{ $(this).hide()}); // 商品元素渐隐消失
86         ggroup.prepend(item);                               // 添加商品到购物车
87         item.animate({ opacity: 1 }, 500);                  // 购物车商品渐显
88     }
89 });
90 </script></html>
```

9.6.3 代码分析

本例为了使代码更加简洁，使用第三方类库jQuery开发，官网地址为http://jquery.com。

代码第09行~第21行给出了单个商品列表元素的DOM结构。需要注意，在最外围的li元素中设置draggable属性为true，表示该元素允许被拖曳，可以被用来触发drop事件。

代码第24行~第29行为购物车的DOM结构，该区域样式被设置为一直停留在浏览器的右下角。

代码第30行~第36行，使用script标签存储购物车商品元素的HTML字符模板，注意脚本

标签的type属性，这里被赋予“text/x-html5-tmpl”的自定义内容，浏览器不会将该模板内容识别为脚本并执行。



提示 除了使用自定义脚本类型的方法存储HTML字符模板外，通常还会使用textarea标签进行存储，这是前端开发中非常实用的技巧，读者可以运用在平常开发中。

结束了DOM结构的分析，下面进入本示例的关键JavaScript部分。

代码第46行~第50行，监听购物车底部红色横条的单击事件，实现购物车列表的收缩和展开功能。这里使用jQuery的animate方法实现动画效果，语法如下：

```
element.animate( properties [,duration] [,easing] [,complete] )
```

- properties：CSS属性对象，对应的值表示属性的最后状态。
- duration：动画持续时间，默认为400ms，可选。
- easing：缓动函数的类型，可选。
- complete：执行完毕后的回调函数。



提示 animate方法更多信息可以参考网站http://api.jquery.com/animate/。

代码第51行~第60行，监听购物车外围容器的单击事件，当触发事件的目标元素经历事件冒泡，最终被监听事件捕捉后，通过样式类名判断触发元素是否为关闭按钮。如果发现为关闭按钮，则移除按钮对应的购物车商品，并恢复左侧商品列表内对应的商品。

代码第61行~第72行，监听左侧列表的每个商品元素的mouseenter、mouseleave、dragstart、dragleave、dragend事件。监听mouseenterg事件和mouseleave事件，是为了实现移入和移出商品时背景色变换效果。在dragstart事件中，代码第66行是实现拖曳效果必须注意的地方，dataTransfer的dropEffect属性规定了允许拖动的事件行为，与dragover监听事件的移动效果相对应。

代码第73行~第79行，监听购物车的dragover和drop事件。其中，代码第76行设置dataTransfer的dropEffect属性为move，表示当元素被拖曳入其中后对应的效果。当购物车drop事件被触发后，首先获取商品列表中的商品信息，并填入购物车元素HTML字符模板，产生购物车商品元素。接着，将该元素添加到购物车的顶部，并使用动画淡进效果出现，同时被拖入的元素使用动画淡出效果渐渐地消失。



9.7 示例7 拖放图片保存服务器

9.7.1 示例效果

很多网站都会在个人中心模块提供一个相册的功能，用户可以将平日喜爱的照片上传至相册与朋友进行分享交流。本例将从实际应用出发，实现相册中的图片上传功能，程序包含客户端和服务器端，读者可以通过本例的学习，全面完整地了解制作上传图片模块的过程。

使用Chrome浏览器打开网页文件“拖放图片保存服务器.html”，效果如图9.15所示。

打开本例对应的服务器端脚本文件夹“upload-server”，如图9.16所示。“server.js”存放了主要的上传服务逻辑脚本，“package.json”存放了模块的描述信息，“node_modules”文件夹存放程序依赖模块，“images”用于存放客户端上传的图片文件。



图9.15　使用Chrome浏览器打开网页文件



图9.16　服务器端脚本文件夹“upload-server”

Windows下打开命令行进入对应的“upload-server”文件夹，执行代码如下：

```
node server.js
```

如果启动成功，命令行提示如下：

```
listening on http://localhost:8080
```

打开光盘源代码库的“images\product”商品图片文件夹，选中所有图片并拖曳至示例页面“拖曳图片至此”区域，效果如图9.17所示。



读者可以挑选任意其他图片文件夹进行拖曳。



图9.17　选中图片贴拖曳至网页上传区域

松开鼠标，被拖曳的图片显示在网页中，并且每张图片中央都带有一个矩形的上传进度条，效果如图9.18所示。

单击上传图片按钮，图片中央的进度条随着图片的上传变为百分之百，同时服务器端的images文件夹出现刚刚上传的图片，效果如图9.19所示。



图9.18 松开鼠标，图片显示在网页中



图9.19 所有图片上传完毕

9.7.2 代码设计和分析

首先分析客户端网页代码，打开“拖放图片保存服务器.html”，代码如下：

```
<!doctype html><html>
<head><style>/* ...样式代码省略，详见本书网络资源*/ </style><script src="../js/jquery-1.8.3.js"></script></head>
<body>
    <header><h2>拖放图片保存服务器</h2></header>
    <ul class="picture-list">
        <script id="J_item_tpl" type="text/x-html5-tmpl">
        <div class="info">
            <div class="img">
                <div class="wraper">
                    <img src="{src}" title="{name}"/>
                    <div class="progressbar"><div class="progressbar-value"></div></div>
                </div>
            </div>
            <h3 class="picture-title"><a href="#" target="_blank">{name}</a></h3>
        </div>
        </script>
        <div class="picture-tip">拖曳图片至此</div>
    </ul>
    <a href="#" class="button green">上传图片</a>
</body>
<script>
var slice = Array.prototype.slice,              // 获取数组slice原型方法
    tpl = $('#J_item_tpl').html(),     list = $('ul.picture-list'),
    list_tip = $('div.picture-tip'),            upload_btn = $('a.button'),
    substitute = function (str, sub) {          // 字符串格式化函数
```

```
            return str.replace(/\{(.+?)\}/g, function ($0, $1)
{ return $1 in sub ? sub[$1] : $0; });
        },
        file_list = [];
function render_list(files) {
        file_list = file_list.concat(files);
        files.forEach(function (file, index) {        // 循环File数组
            var reader;                         // reader变量存放FileReader实例
               item = $(document.createElement('li')).attr("class",
'picture');
            if (file.type.toLowerCase().match(/image.*/)) {
            // 用正则表达式判断文件是否为图片类型
                reader = new FileReader();
                // 实例化FileReader对象，用于读取文件数据
                reader.addEventListener('load', function (e) {
                // 监听FileReader实例的load事件
                    item.html(substitute(tpl, { name: file.name,
 src: e.target.result }));
                    list.append(item);               // 添加到商品列表尾部
                });
                reader.readAsDataURL(file);          // 读取文件为DataURL
                files.length ? list_tip.hide() : list_tip.show();
            };
            item.bind({
                mouseenter: function () { item.addClass('picture-hover'); },
                mouseleave: function () { item.removeClass('picture-
hover'); }
            });
        });
};
function upload_file(file,index) {                   // 用于上传单个文件
        var xhr = new XMLHttpRequest(),// 实例化XMLHttpRequest对象，用于与后台通信
            formData = new FormData() // 实例化FormData对象，用于存储表单数据信息
            item = list.find('li').eq(index),
            progress = item.find('div.progressbar-value');
            // 获取对应上传文件的显示进度条
        formData.append('file', file); // 添加文件数据到formData对象
        xhr.open('post', 'http://localhost:8080', true);
        // 初始HTTP请求，设置请求类型、地址、是否异步
        xhr.addEventListener('load', function () {        // 读取完毕后触发
            var data = JSON.parse(xhr.responseText); // 将返回的字符串转化为对象
            item.find('a').attr('href', 'http://localhost:8080/' +
data.name);
        });
        xhr.upload.addEventListener('progress', function (e) {//读取中触发
            var percent;
            if (e.lengthComputable) {                // 是否可以计算上传字节数
                percent = (e.loaded / e.total * 100) + '%';
                // 已上传字节数除以总字节，计算上传百分比
                progress.css('width', percent);
           // 更新上传进度条样式为当前上传比例相同
```

```
65                progress.html(percent);           // 更新上传百分比
66                file_list[index] = null;          // 清空列表对应索引内容
67            };
68        });
69        xhr.send(formData);                         // 发送数据到服务器
70        progress.css('display', 'block');
71    };
72    list.bind('dragover', function (e) { e.preventDefault();});
      // 拖曳在该元素上移动时触发
73    list.bind('drop', function (e) {         // 拖曳在该元素上释放时触发
74        e.preventDefault();
75        render_list(slice.call(e.originalEvent.dataTransfer.files, 0));
76    })
77    upload_btn.bind('click', function (e) {
78        e.preventDefault();
79        file_list.forEach(function (item, index) {
80            item && upload_file(item, index);  // 文件如果存在，则上传至服务器
81        });
82    });
83    </script></html>
```

代码第06行~第16行，script标签内存放了上传后图片元素的HTML字符模板，用script标签存放模板是前端开发中的常用技巧。

代码第29行~第48行为函数render_list，用于读取图片文件，生成上传图片列表。代码第31行，将每次拖曳的图片文件对象都存入变量file_list中，以便在上传时方便地获取图片文件对象。循环开始，生成样式类名为picture的li元素作为上传图片容器。通过File对象的type属性判断文件是否为图片类型。对于图片类型文件，新建一个FileReader的实例，调用实例方法readAsDataURL读取文件内容，在监听读取文件对象的load事件回调函数中，组装上传图片元素的字符模板并添加到页面图片列表元素的末尾。

代码第49行~第71行为函数upload_file，用于将图片文件对象通过Ajax返回给图片服务器进行保存。代码第52行，通过传入参数index，获取对应列表位置的图片容器元素。将文件对象填充入formData的实例，设置XMLHttpRequest的实例对象的远程请求地址为http://localhost:8080，该上传地址即通过Node.js实现的图片上传服务器地址。图片上传完毕后，将file_list数组对应的index索引位置内容设置为null，确保该图片文件不被二次上传。

代码第80行，每次单击上传图片按钮，循环file_list数组并调用upload_file方法，在调用之前会先判断数组索引位置内容是否为空。

下面分析服务器端代码。服务器端逻辑采用Node.js实现，提供了两部分功能：查看图片类型请求和保存上传内容。

先看查看图片的GET请求实现，代码如下：

```
var pathname = url.parse(req.url).pathname,    // 从请求地址分析文件名
    filepath = _dirname + '\\images' + pathname;      // 图片文件地址
fs.readFile(filepath, "binary", function (err, file) {
// 读取文件为binary
    if (err) {
        res.writeHead(500, { 'Content-Type': 'text/plain' });
        // 读取错误并返回500头信息
```

```
        res.end('500');
    } else {                               // 设置正确的Content-Type信息
        res.writeHead(200, { 'Content-Type': 'image/' +
path.extname(pathname).slice(1) });
        res.write(file, "binary");         // 写入图片二进制流内容
        res.end();
    }
});
```

变量“_dirname”为Node.js全局变量，表示服务脚本的具体运行时地址。fs变量为对应File System模块的引用，用于文件的操作。fs对象的readFile方法用于读取文件内容，语法如下：

```
fs.readFile(filename, [options], callback)
```

- filename：文件具体的地址。
- options：文件读取的编码格式。
- callback：读取结束后的回调函数，回调函数接收两个参数，错误信息和文件内容。

接着分析上传保存部分脚本逻辑，代码如下：

```
var single_file, file_name;
form.on('end', function () {                      // 监听form上传结束事件
    res.write(JSON.stringify({"code": 200,  "name": file_name }));
                  // 写入上传成功状态码和信息
    res.end();                                    // 响应结束
}).on('file', function (field, file) {            // 监听form文件上传事件
    single_file = file;                     // 将上传文件file实例存入数组
    file_name = crypto.createHash('md5').update(single_file.name.
split('.')[0]).digest('hex');
    file_name += '.' + single_file.name.split('.')[1];
}).on('error', function (error) {                 // 监听form上传出错事件
    res.write(JSON.stringify({"code": 500,  "msg": '上传失败，'
+ error })); // 写入上传失败状态码和信息
    res.end();
});
form.parse(req, function (err) {                  // 解析请求信息并保存文件
    util.pump(fs.createReadStream(single_file.path),
      // 读取文件内容并写入地址
            fs.createWriteStream(__dirname + '\\images\\' + file_
name), function (err) {
                if (err) { console.log(err);  }
            });
});
```

变量form为第三方模块formidable中IncomingForm的实例，代码如下：

```
var formidable = require("formidable");    // 引用formidable模块，用于文件保存
var form = new formidable.IncomingForm();  // 新建IncomingForm实例
```

在file事件的监听函数中，将上传文件的文件名进行MD5加密处理，防止因中文文件名导

拖动脚本文件压缩

```
var
Words=Collection.extend({constructor:function(scri
pt)
{this.base();forEach(script.match(WORDS),this.add,
this);this.encode()},add:function(word)
{if(!this.has(word))this.base(word);word=this.get(
word);word.count++;return word},encode:function()
{this.sort(function(word1,word2){return
word2.count-word1.count});eval("var
a=62,e="+Packer.ENCODE62);var encode=e;var
encoded=new Collection;var
count=this.size();for(var i=0;i<count;i++)
{encoded.put(encode(i),i)}var empty=function()
{return""};var index=0;forEach(this,function(word)
{if(encoded.has(word))
{word.index=encoded.get(word);word.toString=empty}
```

图9.22 脚本文件被压缩显示

9.8.2 代码设计

利用编辑器打开“拖动脚本文件压缩.html”文件，代码如下：

```
<!doctype html>
<html>
<head>
     <!-- 开源脚本压缩工具 -->
     <script src="../js/base2-load.js"></script><script src="../js/Packer.js"></script>
     <script src="../js/Words.js"></script>
     <style>
     textarea{                                /* 文本框样式 */
        font-family: 宋体; width:320px; height:220px;
        border: 2px solid black; color: #777;
        box-shadow: 0 0 5px #ddd inset;        /* 文本框阴影 */
        margin: 0px auto; border-radius: 2px;
        background-color: #F9F9F9; font-size: 12px;
     }
     </style>
</head>
<body>
    <header><h2>拖动脚本文件压缩</h2></header>
    <textarea>                                <!-- 文本区域 -->>
    // JavaScript在线压缩工具
    // 请将代码文件拖曳到此处
    </textarea>
</body>
<script>
    (function () {
        var textarea = document.querySelector('textarea');
        textarea.addEventListener('dragover', function (e) {
        // 拖曳在该元素上移动时触发
            e.preventDefault();        // 阻止元素默认事件，确保drop正常触发
        }, false);
        textarea.addEventListener('drop', function (e) {
            // 鼠标拖曳在该元素上释放时触发
            e.preventDefault()                        // 阻止元素默认事件
```

自定义本地文本阅读器的可能。本例将展现一个相对简单的本地阅读器版本，读者可以自行修改代码实现更加丰富的功能。使用Chrome浏览器打开网页文件，运行结果如图9.23所示。

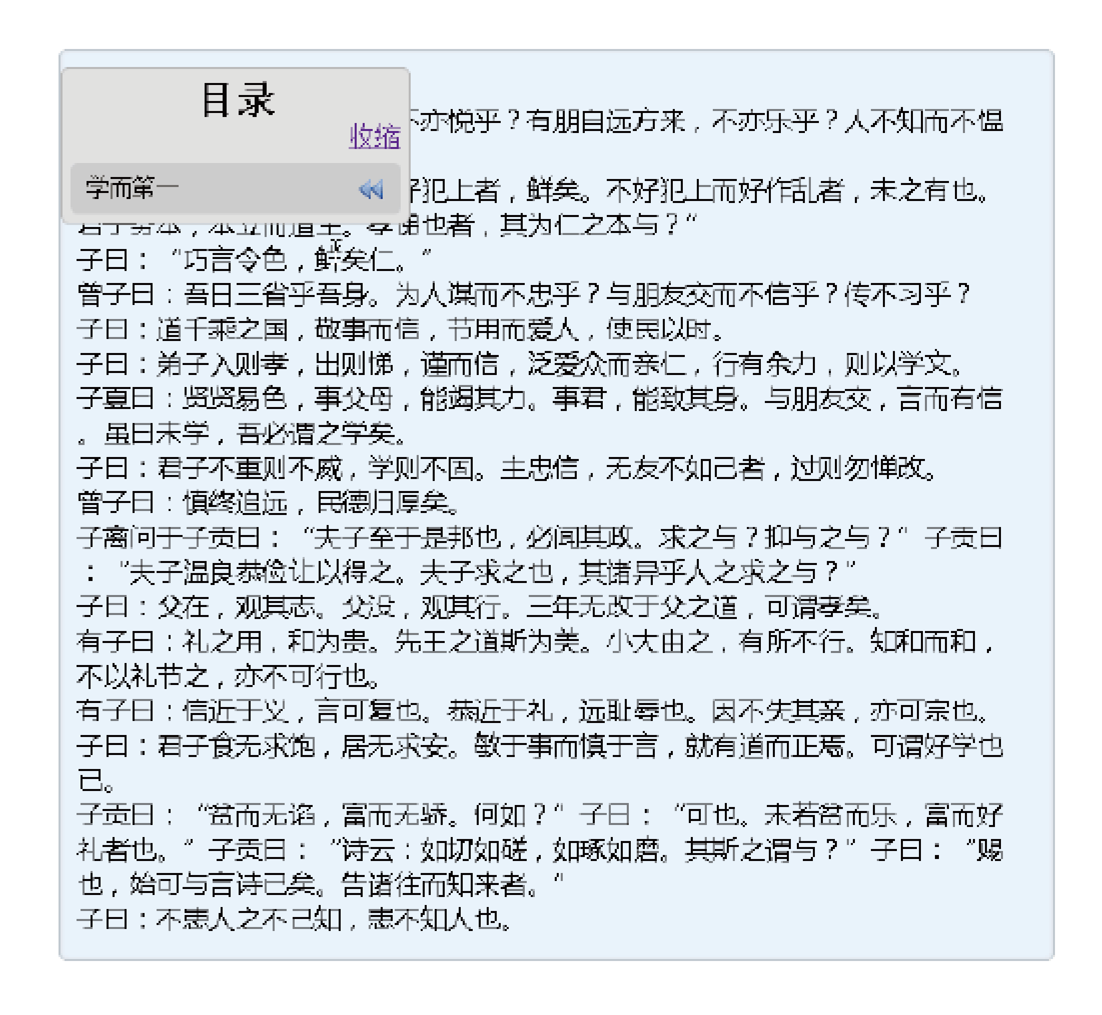


图9.23　使用Chrome打开网页文件

选中本地文件夹中的文本章节，拖曳文本章节至“目录”区域，效果如图9.24所示。

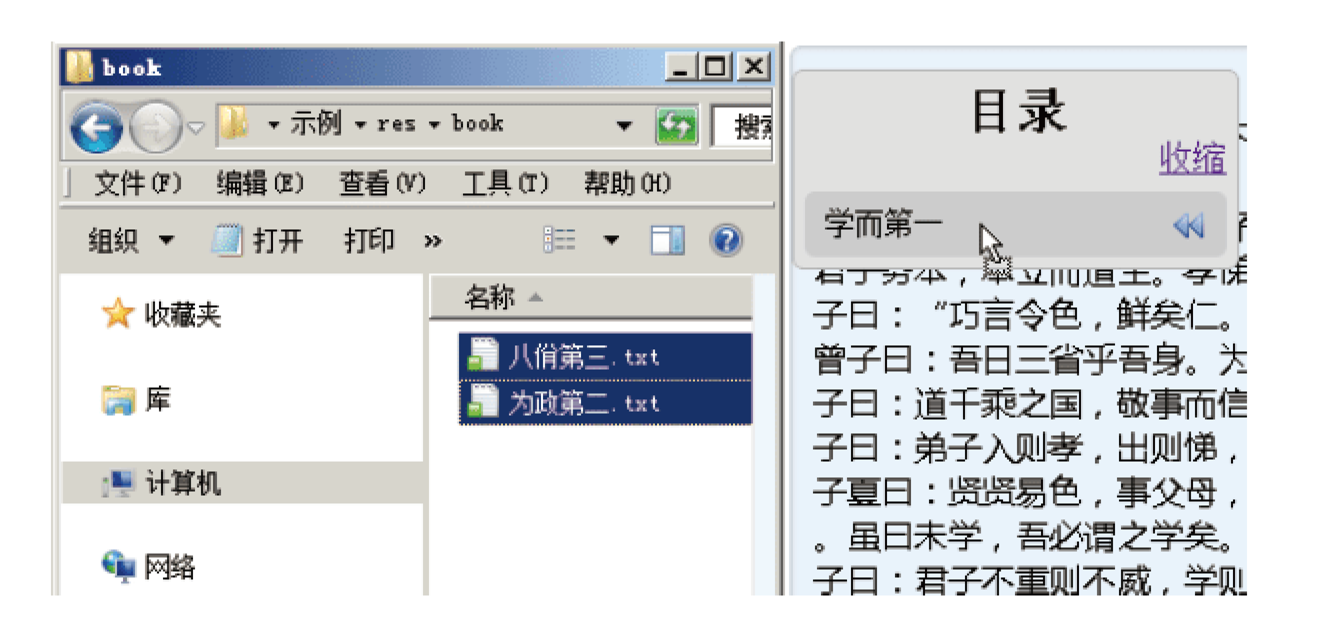


图9.24　选中本地文本拖曳至“目录”区域

松开鼠标，“目录”区域新增添加对应的章节文件导航，效果如图9.25所示。

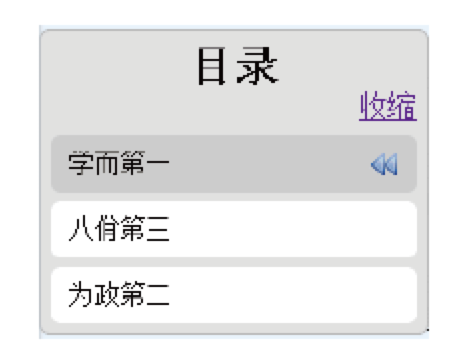


图9.25　新增拖曳章节导航

单击“为政第二”章节导航，阅读器内容随即切换为该章节内容，效果如图9.26所示。

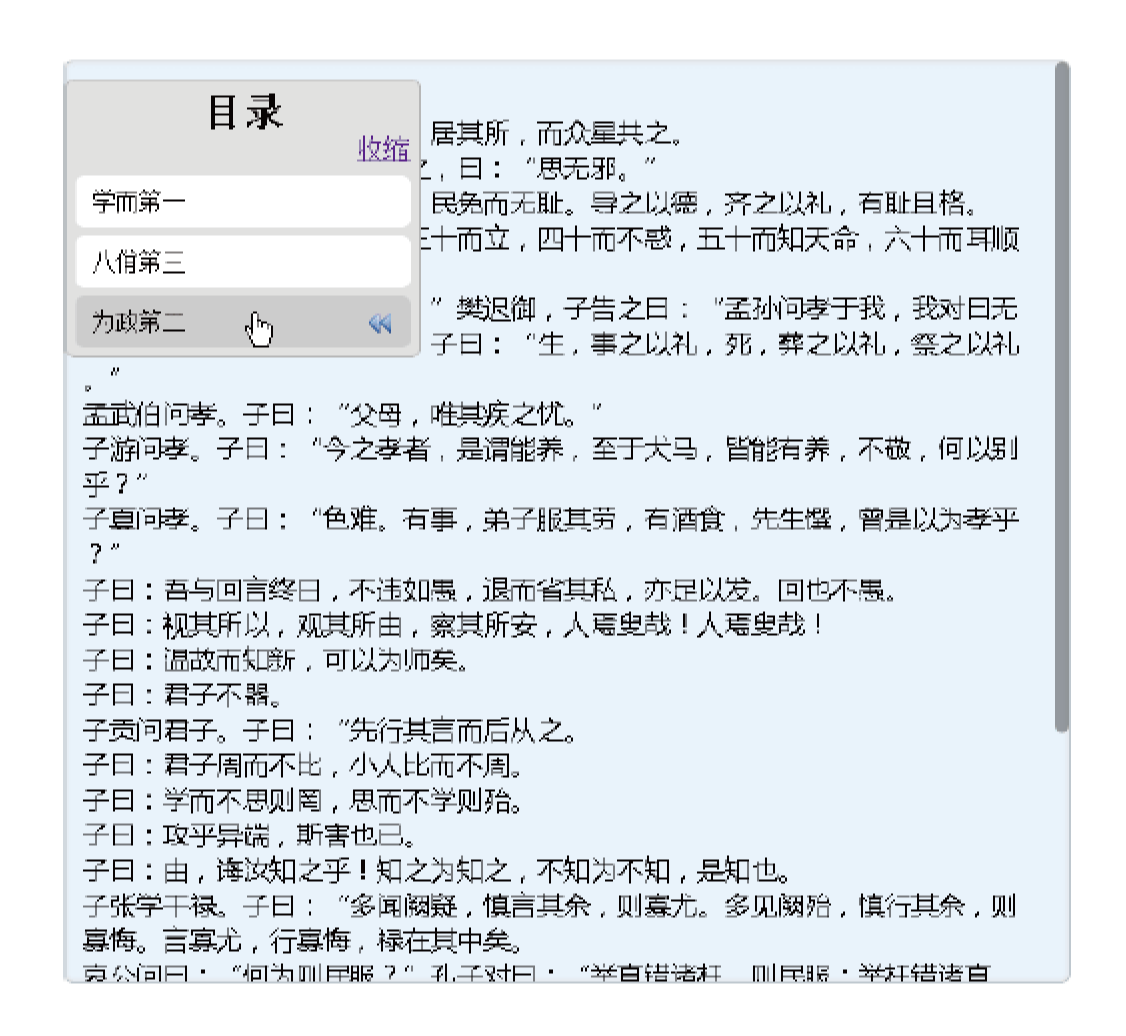


图9.26 单击“为政第二”章节导航

该阅读器还具备收缩目录导航功能，单击“目录”区域“收缩”按钮，该区域收缩至阅读器外侧左上角，效果如图9.27所示。

图9.27 单击“收缩”按钮

9.9.2 代码设计

利用编辑器打开“可拖放文本阅读器.html”文件，代码如下：

```
<!DOCTYPE HTML><html>
<head><style>/* ......省略样式代码，详见本书网络资源 */</style></head>
<body>
    <header><h2>可拖放文本阅读器</h2></header>
    <section>
        <aside class="bar"><h2>目录</h2></aside><!-- 文章目录缩小区域 -->
        <aside class="list">                    <!-- 文章标题列目录区域 -->
```

```
                <h2>目录</h2>
                <div class="handle"><a href="#">收缩</a></div>
                <div class="title active">学而第一</div>
            </aside>
            <article>/* ......省略文章内容，详见本书网络资源 */</article>
<!-- 文章内容区域 -->
        </section>
</body>
<script>
var slice = Array.prototype.slice,
    list_bar = document.querySelector('.bar'),       // 目录标签
    list = document.querySelector('.list'),          // 悬浮目录列表
    shrink_btn = list.querySelector('.list a'),      // 缩小按钮
    article = document.querySelector('article'),     // 阅读文本
    first_article = document.querySelector('.title'),// 首个文章标题
    CACHE = {},                                      // 文章缓存变量
    article_click_event = function (e) {             // 单击文章导航事件
        var titles = document.querySelectorAll('.title');
        for (var i = 0, l = titles.length; i < l; i++) {
              // 移除所有当前样式
            titles[i].className = 'title';
        };
        this.className += ' active';                 // 设置单击按钮样式
        article.innerHTML = CACHE[this.innerHTML];   // 获取缓存内容
    };
first_article.addEventListener('click', article_click_event, false);
list_bar.addEventListener('click', function (e) {    // 绑定展开按钮
    list_bar.style.display = 'none';
    list.style.display = 'block';
}, false);
shrink_btn.addEventListener('click', function (e) {  // 绑定缩小化按钮事件
    list_bar.style.display = 'block';
    list.style.display = 'none';
}, false);
list.addEventListener('dragover', function (e) {
      // 拖曳在该元素上移动时触发
    e.preventDefault();
}, false);
list.addEventListener('drop', function (e) {
      // 拖曳在该元素上释放时触发
    e.preventDefault();
    var files = slice.call(e.dataTransfer.files, 0);
    files.forEach(function (file) {
        var reader = new FileReader(); // 实例化FileReader对象，读取文件数据
        reader.addEventListener('load', function (e) {
        // 监听FileReader实例的load事件
            var name = file.name.split('.')[0],
                data = CACHE[name];               // 按文章名获取文章内容
            if (!data) {
                CACHE[name] = e.target.result;       // 缓存文章内容
                var item = document.createElement('div');
```

```
                        // 创建文章标题导航
54                      item.className = 'title';
55                      item.innerHTML = name;
56                      item.addEventListener('click', article_click_event,
    false);
57                      list.appendChild(item);             // 添加到导航条末尾
58                  };
59              }, false);
60              reader.readAsText(file);                    // 将文件读取为文本
61          });
62  }, false);
63  CACHE[first_article.innerHTML] = article.innerHTML;
    // 缓存页面内容
64  </script></html>
```

9.9.3 代码分析

文本阅读器分成目录和内容两块区域。目录区用于拖曳本地文件和展现选择显示的标题，内容区用于展现选中标题的内容。代码第17行~第21行是脚本需要操作的元素节点。代码第22行用于存放当前页面文章和拖曳文章的内容变量对象，对象的键为文本的名称，值为文本文档的内容。

代码第23行~第30行为函数article_click_event，用来处理单击目录文章名后的操作。首先，移除所有导航目录的激活状态，然后给被单击的文章标题添加激活的样式，最后从缓存变量对象CACHE中获取对应标题的文本内容填充至阅读区域。

代码第32行~第39行，主要实现目录的收缩和展开功能。

代码第43行~第62行实现本地文本文件拖曳至阅读器的功能。首先，监听目录列表的drop事件，在触发该事件后，获取拖曳的文件对象，循环使用FileReader对象读取每个文件的内容，关键方法为readAsText，语法如下：

```
fileReader. readAsText(file, encoding)
```

- file：文件对象。
- encoding：返回数据所使用的编码。如果不指定，默认为UTF-8。

文件读取完毕，会触发FileReader对象的load事件。在该事件中，首先，获取文件去除后缀的名称，将获取的文件内容以文件名为键缓存至CACHE变量对象中，同时，创建一个新的DIV元素表示文章的标题，添加至目录末尾。

代码第63行，获取页面原有章节内容缓存至对象变量CACHE中，确保CACHE变量拥有所有章节内容信息。



第10章 本地存储大演练

HTML 5不仅带来了丰富的语义标签，同时还带来了一项非常强大的特性，称为离线本地存储。在Web 2.0时代，客户端的数据存储基本通过Cookie来实现，这给应用带来了极大的限制和网络性能消耗。HTML 5提出了全新的方案，本章示例将介绍SessionStorage、LocalStorage的原生用法，同时还会介绍和分析笔者开发的离线缓存类库源码，深入浅出地让读者了解在不同版本的浏览器之间实现统一接口的离线缓存方案，最后还会介绍如果安全地使用本地存储。

本章知识点：

- SessionStorage和LocalStorage
- 制作一个桌面提醒工具
- 各种离线缓存方案



10.1 示例1 保存与读取登录用户名与密码

10.1.1 示例效果

日常的登录网页，在现在的浏览器中已经具备将每次登录成功的信息进行记录的功能，用户可以在下次登录时，轻松地输入部分内容并补充完整信息。本例从实际出发，采用HTML 5本地存储功能模拟浏览器这个过程。

使用Chrome浏览器打开网页文件，运行效果如图10.1所示。在“昵称”输入框和“密码”输入框内输入测试用户信息，昵称和密码均为test，单击“登录”按钮，运行效果如图10.2所示。



图10.1 使用Chrome打开网页文件



图10.2 输入信息单击“登录”按钮

登录后，“昵称”和“密码”输入信息消失。单击“昵称”输入框，下方出现下拉账户提示，内容为刚刚输入的“昵称”信息，效果如图10.3所示。单击提示框内test账号，提示框消失，“昵称”和“密码”输入框内被填入内容，效果与图10.2相同。

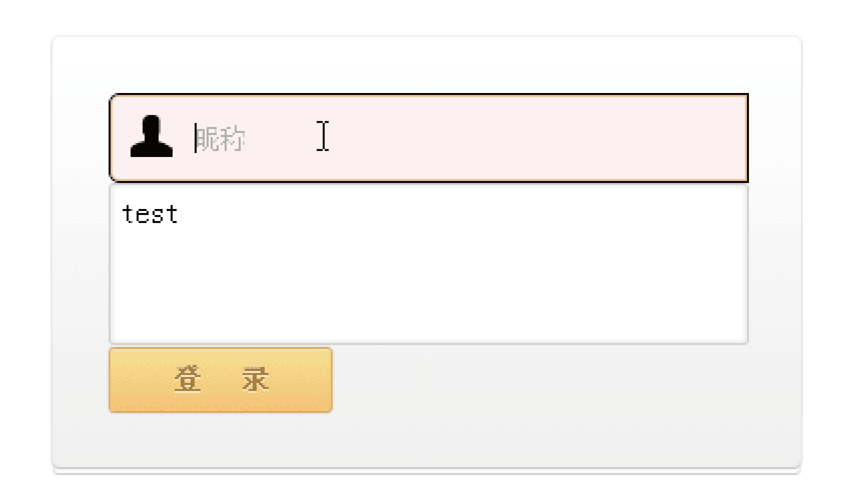


图10.3　“昵称”输入框下方出现账户提示信息

10.1.2 代码设计

利用编辑器打开“保存与读取登录用户名与密码.html”文件，代码如下：

```
<!doctype html><html>
<head>
    <style>/* ......省略样式代码，详见本书网络资源 */</style>
    <script src="../js/jquery-1.8.3.js"></script>
            <!-- 本例依赖于jQuery -->
</head>
<body>
    <header><h2>保存与读取登录用户名与密码</h2></header>
    <section>
        <form action="" method="post">
            <div class="clearfix first">   <!-- 去除输入框默认下拉提示 -->
                <input type="text" tabindex="1" id="name" class="item-
name" placeholder="昵称"
autocomplete="off" required/>
                <ul class="suggest"></ul>     <!-- 下拉提示区域 -->
            </div>
            <div class="clearfix">       <!-- 去除输入框默认下拉提示 -->
                <input type="password" tabindex="2" id="password"
class="item-password" placeholder="密码" autocomplete="off" required/>
            </div>
            <div class="clearfix"><input type="submit" tabindex=
"3" id="submit" value="登  录" /></div>
        </form>
    </section>
</body>
<script>
    var el_name = $('#name'),                        // 昵称输入框
        el_password = $('#password'),                // 密码输入框
        el_submit = $('#submit'),                    // 登录按钮
        el_suggest = $('ul.suggest'),                // 下拉提示容器
```

```
28          el_form = $('form'),                              // 表单
29          storage_key = 'html5_local';                      // 缓存key
30      el_suggest.on('click', function (e) {
31          var target = $(e.target);
32          el_name.val(target.text());                       // 写入昵称
33          el_password.val(target.attr('data-pass'));        // 写入密码
34      });
35      el_form.on('submit', function () {            // 截取表单提交保存账号密码
36          var data = localStorage.getItem(storage_key);
            // 获取key缓存内容
37          data = data ? JSON.parse(data) : {};
38          data[el_name.val().trim()] = el_password.val().trim();
            // 缓存账号和密码
39          localStorage.setItem(storage_key, JSON.stringify(data));
            // 设置缓存
40      });
41      var storage_data = localStorage.getItem(storage_key),
        // 读取缓存
42          html_arr = [];                                    // html字符数组
43      if (storage_data) {
44          storage_data = JSON.parse(storage_data);          // 解析为对象
45          for (var key in storage_data) {                   // 循环缓存对象
46              html_arr.push('<li data-pass="' + storage_data[key] +
   '">' + key + '</li>');
47          };
48          el_suggest.html(html_arr.join(''));               // 设置下拉框结构
49          el_name.on({
50              'mousedown': function () { el_suggest.fadeIn(); },
                     // 单击展示下拉框
51              'blur': function () { el_suggest.fadeOut(); }
                     // 失去焦点时隐藏下拉框
52          });
53      };
54  </script>
55  </html>
```

10.1.3 代码分析

本例的存储功能使用HTML 5的LocalStorage实现，代码第29行的变量storage_key被用于本地存储的缓存键值。

第30行~第34行代码，监听下拉提示列表的单击事件，当用户单击下拉框内账户信息时，从对应的账户DOM元素节点中获取昵称和密码，昵称为元素内本文节点内容，密码为元素自定义属性“data-pass”的值。

第35行~第40行代码，监听表单提交事件，当用户单击“登录”按钮提交表单内容，首先调用LocalStorage的getItem方法获取键值为storage_key的本地存储的内容，该方法语法如下：

```
localStorage.getItem(key)                    // 获取键值为key的值，key为字符型
```

获取完毕后，调用JSON的parse方法将字符串解析为JavaScript对象字面量，并存入表单的昵称和密码数据，然后调用LocalStorage的setItem方法进行本地存储，该方法语法如下：

```
localStorage.setItem(key,data);          // key为键，data为值，均为字符型
```

LocalStorage存储的对象必须为字符型，所以在存储前，将JavaScript对象调用JSON的stringify方法进行字符序列化。

代码41行~第53行做了两件事情，首先获取本地存储的昵称和账号信息生成下拉提示框HTML结构，然后监听“昵称”输入框的mousedown和blur事件，实现提示框出现和消失的逻辑。



10.2 示例2 保存与读取临时数据

10.2.1 示例效果

本例将使用HTML 5的SessionStorage，为第9章中的“拖放图标.html”示例添加图标位置记忆功能。用户可以拖曳页面图标，即使刷新页面，图标仍然停留在拖曳后的位置，只有把页面关闭后重新打开，图标才重置回原位。

使用Chrome浏览器打开网页文件，运行效果如图10.4所示。随意拖曳页面上各个图标，效果如图10.5所示。



图10.4 使用Chrome打开网页文件



图10.5 拖曳页面各图标

按F5键刷新页面，图标仍然停留在最后拖曳的位置，效果与图10.5相同。关闭该页面，并重新打开页面，图标被重置为原位，效果与图10.4相同。

10.2.2 代码分析

拖曳部分的实现代码可以参考“拖放图标”示例的讲解，本例关注在临时数据存储这块，下面看看具体有哪些改动。

10.3.2 代码设计和分析

本例引用第三方类库jQuery和jQuery UI来简化代码的开发和减少代码的数量。首先，看逻辑脚本初始区代码，代码如下：

```
01 var   DB_NAME = 'html5_storage_form_test',     // 数据库
02       item_tpl = $('#J_item').html(),          // 列表中的单个元素模板
03       substitute = function (str, sub) {       // 字符串格式化函数
04              return str.replace(/\{(.+?)\}/g, function ($0, $1)
   {return $1 in sub ? sub[$1] : $0;  });
05       },
06       tbody = $('table.data tbody'),                  // 列表容器
         // 打开WebSQL数据库，不存在则创建
07       storageDriver = window.openDatabase(DB_NAME, '1.0',
    'html5 storage table', 1048576);
```

代码第01行，申明变量DB_NAME并赋予值“html5_storage_form_test”，表示数据库名称。第03行~第05行，定义方法substitute，用于格式化字符串。代码第07行，使用openDatabase方法打开一个本地数据库，如果数据库不存在则进行创建，语法如下：

```
window.openDatabase(name,version,displayName,estimatedSize,DatabaseCallback)
```

- name：数据库名称。
- version：版本号，目前版本均为1.0。
- displayName：对数据库的描述。
- estimatedSize：数据库预设大小。
- DatabaseCallback：回调函数，可省略。

接下来，看两个方法add_item和del_item，分别用于向列表添加和删除一条内容，代码如下：

```
function add_item(data, toDB) {                     // 添加列表项、数据库
    if (toDB) {
        storageDriver.transaction(function (t) {   // 往数据库插入一条数据
            t.executeSql("INSERT INTO " + DB_NAME + " (name,sex,age)
 VALUES (?,?,?);",
                [data.name, data.sex, data.age],     // 传入保存数据
                function (transaction, resultSet) {
                    data.id = resultSet.insertId; // 获取数据库返回的自增ID
                    tbody.append(substitute(item_tpl, data));
                    // 添加到列表
                }, function () { alert('添加失败'); });  // 错误回调函数
        });
    } else { tbody.append(substitute(item_tpl, data)); };
    // 添加元素到列表
};
function del_item(target) {                                // 删除一条数据
    storageDriver.transaction(function (t) {               // 执行删除SQL
        t.executeSql("delete from " + DB_NAME + " where id=" +
 target.attr('data-id'),
            [], function () {
```

```
17                  target.closest('tr').fadeOut();
                    // 隐藏元素
18              });
19          });
20  };
```

方法add_item接收两个参数，第一个用于传递行内数据，第二个用于决定是否将数据存入本地数据库。代码第03行，调用storageDriver对象的transaction方法创建一个事务，该方法语法如下：

```
database.transaction(callback,errorCallback,successCallback)
```

- callback：事务创建成功后，回调函数返回一个该事务对象。
- errorCallback：事务创建失败后，回调函数返回一个错误对象，可选。
- successCallback：事务创建成功后回调，无返回数据，可选。



使用过传统数据库的读者对事务应该不会陌生，事务是以执行过程中的一个逻辑为单位，可以控制一系列操作。想了解更多的事务说明，详见网址http://zh.wikipedia.org/wiki/数据库事务。

代码第04行~第09行，调用事务对象的executeSql方法执行1条插入功能的SQL语句，executeSql方法语法如下：

```
transaction. executeSql(sqlStatement, arguments, callback, errorCallback)
```

- sqlStatement：SQL字符串。
- arguments：传入给SQL字符串对应问号处的参数数据，数组类型。
- callback：执行成功回调，返回两个参数，事务对象和数据结果。
- errorCallback：执行失败回调，返回两个对象，事务对象和错误对象。

其中的第07行代码，获取返回结果对象的insertId属性，并存入数据对象变量data。insertId为成功插入数据库后，返回的自增ID字段的值。

方法del_item接收一个元素节点参数，用于执行在本地数据库中删除对应节点ID的内容，每行对应的ID被保存在自定义属性data-id中。

下面看看如何创建列表功能所需的数据库表结构，代码如下：

```
storageDriver.transaction(function (t) {      // 启动一个事务生成列表
    t.executeSql("CREATE TABLE IF NOT EXISTS " + DB_NAME +
    // 创建数据表
                "(id INTEGER PRIMARY KEY AUTOINCREMENT, " +// 自增字段
                    "name TEXT NOT NULL, " +              // 姓名字段
                    "sex TEXT NOT NULL, " +               // 性别字段
                    "age INTEGER DEFAULT 0)");            // 年龄字段
    t.executeSql("SELECT * FROM " + DB_NAME, [],          // 读物数据表
        function (t, results) {
            for (var i = 0, l = results.rows.length; i < l; i++) {
                add_item(results.rows.item(i), false);    // 生成列表
            };
        });
});
```

上面的代码一共执行了两条SQL语句，首先看第一条，用于创建用户信息的数据表，表结构如下。

- id：数值型的自增主键。
- name：表示昵称，文本型非空。
- sex：表示性别，文本型非空。
- age：表示年龄，数字型，默认为0。

第二条SQL语句，用于查询数据库表中的内容。当查询成功后，通过结果对象results的rows属性返回数据，并通过调用add_item方法，在列表中创建行元素结构。

继续看每行操作区域删除功能的实现，代码如下：

```
01 tbody.on('click', function (e) {                  // 监听列表单击事件
02     var target = $(e.target);
03     if (target.hasClass('del')) {                 // 目标元素为删除按钮
04         e.preventDefault();                       // 阻止a元素默认事件
05         del_item(target);                         // 删除元素和数据库内容
06     };
07 });
```

为了避免每次创建列表行元素时监听对应删除按钮的单击事件，将监听事件绑定到列表的tbody元素上，如此每次单击“删除”按钮，都能通过事件冒泡的形式向外传递并被接收。上方代码第03行，通过样式类名中是否含有del字符判断目标元素是否为删除按钮，当确定后，调用“del_item”方法，删除列表和数据库中对应的行内容。

最后分析列表添加功能的实现，代码如下：

```
01 $('span.add').on('click', function (e) {                 // 添加操作按钮事件
02     var wraper = $($('#J_form').html());                  // 新建添加弹层
03     $(document.body).append(wraper);                      // 注意页面结构
04     wraper.dialog({
05         position: 'center', title: '添加', modal: true,   // 浮层配置信息
06         buttons: [{ text: "确认",
07             click: function () {
08                 var name = wraper.find('input.name').val(),    // 姓名值
09                     sex = wraper.find('select.sex').val(),     // 性别值
10                     age = wraper.find('input.age').val();      // 年龄值
11                 if (name.length && sex.length && age.length) {
12                     add_item({ name: name, sex: sex, age: age }, true);
13                     $(this).dialog("close");                  // 关闭弹出
14                 } else { alert('请正确填写所有填写内容'); }; // 错误验证提示
15             }
16         }, { text: "取消",
17           click: function () { $(this).dialog("close"); }     // 关闭按钮事件
18         }]
19     });
20     wraper.dialog('open');                                // 打开弹出框
21 });
```

代码第02行~第03行，获取页面中浮层字符HTML模板，并添加到页面DOM结构中。

代码第04行~第19行，使用jQuery UI中的Dialog组件初始化浮层元素，同时监听确认和取消按钮的单击事件。在确认按钮的单击事件中，获取浮层中“姓名”、“性别”、“年龄”这三项的值，判断如果均不为空的情况下，调用add_item方法添加到列表并且存入本地数据库。



更多jQuery UI的Dialog组件使用可以参考网站http://jqueryui.com/dialog/。



10.4 示例4 桌面提醒工具

10.4.1 示例效果

本例将结合Chrome浏览器Desktop Notification功能和HTML 5本地存储LocalStorage，制作桌面消息提醒工具。设计该工具来功能时，需要考虑用户是否确切的看到提醒信息。这里假设是通过单击桌面提醒框来判定该条提醒已被阅读，之后不再对用户提醒，并且刷新页面后对应的提醒告知消失。反之，提醒框在3s后自动消失，则认为用户仍然需要提醒，刷新页面后该条提醒保留。

使用Chrome浏览器打开网页文件，列表信息分为三列：“信息”列用于存放桌面提醒框的内容，“时间间隔（秒）”表示提醒框出现的时间间隔，“距下次提醒时间（秒）”是一个动态更新变化的数值，运行效果如图10.10所示。

信息	间隔时间(秒)	距下次提醒时间(秒)
提示信息1	10	10
提示信息2	20	20

图10.10 使用Chrome打开网页文件

每隔1s，“距下次提醒时间（秒）”列数值减1，当数值减少至0时，操作系统右下角出现提醒框信息，效果如图10.11所示。



图10.11 操作系统右下角出现提醒框信息

假使不单击出现的提醒框，其将会在3秒以后消失，列表中对应的第三列计数恢复原值，并重新开始计时，重复先前的效果。当单击系统右下方出现的提醒框时，其会立即消失，列表对应的第三列停止更新，值始终保持为0，刷新页面后列表该条提醒信息消失，效果如图10.12所示。

信息	间隔时间(秒)	距下次提醒时间(秒)
提示信息2	20	18

图10.12 单击第一条提醒框后刷新页面代码设计

10.4.2 代码设计和分析

本环节将针对分析实现脚本逻辑代码，其余部分读者可以参看本书网络资源。首先看初始执行脚本，代码如下：

```
01 var storage_key = "notification_msg",                    // 本地存储缓存键值
02     storage_data = localStorage.getItem(storage_key);
                              // 获取本地存储提醒数据
03 if (storage_data) {
04     storage_data = JSON.parse(storage_data);             // 解析字符串为对象
05 } else {
06     storage_data = [{ msg: '提示信息1', time: 10 }, { msg: '提示信息2',
   time: 20}]; // 默认的临时提醒数据
07     localStorage.setItem(storage_key, JSON.stringify(storage_data));
                 // 缓存默认数据
08 };
09 renderList(storage_data);                                // 构建提示列表
```

先申明一个变量“storage_key”并赋予自定义键值“notification_msg”，再通过LocalStorage的getItem方法，获取该键值本地存储数据。当页面为首次打开时，获取的键值存储数据为空，此时添加默认数据并存入本地离线存储（见代码第06行~第07行）。当再次进入页面时，将键值获取的缓存数据解析为“JavaScript”对象并赋予storage_data变量。最后调用renderList方法，构建提示列表和执行提醒计时器。

下面看renderList方法，代码如下：

```
function renderList(data) {                              // 构建列表
    var htmls = [];
    data.forEach(function (item, index) {
        item.id = index;                                 // 给提醒数据添加索引
        htmls.push(substitute(item_tpl, item));
        // 将组装的结构添加到数组变量中
        item.interval = setInterval(function () {      // 设置计时器
            var last;
            if (!item.ondisplay && !item.isremove) {
            // 非显示状态和非移除状态
                                                     // 获取剩余间隔时间元素
                last = document.querySelector('tr[data-id="' + index +
 '"] :nth-child(3n)'),
                time = parseInt(last.innerHTML) - 1;
                 // 计时减1秒
                if (time === 0) { notification(item); }
```

storage_data添加键值为索引值和值为商品信息的数据。在代码第15行处，调用Storage的setItem方法将增加后的storage_data对象数据存入本地离线缓存，该方法语法如下：

```
Storage.setItem(key, data, isJSON);
```

- key：表示本地离线缓存的键值
- data：数据信息，对象或数组
- isJSON：表示是否获取的值为JSON对象，布尔型

最后在购物车的删除事件中，完成清除本地离线缓存对应商品信息，代码如下：

```
delete storage_data[target.attr('data-index')]
Storage.setItem(storage_key, storage_data, true);
```



10.6 示例6 封堵数据泄漏

10.6.1 示例效果

笔者经常在网上订酒店或宾馆时，喜欢打开多个浏览器标签进行综合比对，想从中选择一家性价比最高的进行预订。首先，被挑中的是一家价格适中并且评价也不错的酒店，此时将其加入当前标签页购物车并准备下单购买。但同时在另外一个标签页中发现一家价格较低但评价一般的酒店，由于笔者手头较紧想省点钱，遂将其加入该标签页购物车。就在快提交的时候反悔了，心想难得订一次，应该要稍微要住的好一点。所以，又切换回之前的标签页，并提交购物车内容。情况出现了，订单中居然出现的是价格低的那一家酒店，这种情况就是通常说的"数据泄漏"。本例将会模拟该场景的发生，同时给出一个使用HTML 5的SessionStorage功能的优化方案。

将示例文件放置于一台Web服务器上，如Apache或IIS，使用Chrome浏览器打开文件URL地址。页面的左侧是采用传统Cookie实现的购物车，右侧是采用HTML 5的SessionStorage功能实现的购物车，运行效果如图10.15所示。

Cookie



SessionStorage



图10.15 使用Chrome打开网页文件

假设刚刚打开的页面为1号标签页，此时新建一个浏览器标签打开相同示例URL地址（即2号标签页）。选择1号标签页的左右两“购物车”内的“宁夏特级枸杞子”，然后选中2号标签页中两购物车的“广西特大罗汉果”，效果如图10.16和图10.17所示。



图10.16 1号标签页



图10.17 2号标签页

此时，单击1号标签页内Cookie购物车购买按钮，“数据泄漏”产生，选中的结果由“宁夏特技枸杞子”变为“广西特大罗汉果”，并弹出“广西特大罗汉果,74”提示，效果如图10.18所示。



图10.18 单击1号标签页内Cookie购物车购买按钮

单击1号标签页内SessionStorage购物车“购买”按钮，提示信息与选中信息维持不变。

原因不难发现，Cookie跟着域名走不受标签的限制，而SessionStorage只作用于当前标签。

10.6.2 代码设计

本例有主次页面两块构建而成，之间通过iFrame调用，主页面代码如下：

```
<!doctype html><html>
<head>                                                /* 布局样式 */
    <style>section{float:left; margin-left:20px;} iframe{height:215px;
    width:260px}</style>
</head>
<body><header><h2>封堵数据泄漏</h2></header>
    <section><h3>Cookie</h3>                          /* Cookie版购物车*/
        <iframe src="006.cookie-storage-cart.html?type=cookie"></iframe>
    </section>
    <section>
        <h3>SessionStorage</h3>             /* SessionStorage版购物车 */
        <iframe src="006.cookie-storage-cart.html?type=storage"></iframe>
</section></body></html>
```

主页面引用了两次cookie-storage-cart.html作为子页面，区别在于传递的type参数不同，不同的参数会触发子页面不同的逻辑过程，下面看子页面的逻辑，代码如下：

```
<!doctype html><html>
<head>
    <style>header h2{ float:left;} header button{ float:right; }</style>
    <link rel="stylesheet" href="../css/cart.css" type="text/css" />
               /* 购物车样式包 */
    <script src="../js/jquery-1.8.3.js"></script>    /* 依赖jQuery */
</head>
<body>
    <header><h2>购物车</h2><button>购买</button></header>
    <ul class="product-list">/* ......省略商品列表HTML结构，详见本书
网络资源 */</ul>
</body>
<script>
    function getQueryString(name) {            /* 获取URL参数数据 */
        var reg = new RegExp("(^|&)" + name + "=([^&]*)(&|$)", "i");
        var r = window.location.search.substr(1).match(reg);
        if (r != null) return unescape(r[2]); return null;
    };
    function getCookie(sName) {                /* 获取指定Cookie值 */
         var sRE= "(?:;.)?" + sName + "=([^;]*);?",
oRE= new RegExp(sRE);
         if (oRE.test(document.cookie))  return decodeURIComponent
(RegExp["$1"]);
         else return null
    };
    function setCookie(name, value) {                /* 添加Cookie */
```

```
        document.cookie = name + "=" + encodeURIComponent(value) + "; path=/";
    };
    var type = getQueryString('type'),          // URL中参数type的值
        list_items = $('li.product'),           // 商品元素列表
        btn_buy = $('button'),                  // 购买按钮
        cache_key = 'select_index';             // 缓存键值
    list_items.bind({                           // 绑定商品列表若干事件
        'mouseenter': function () { $(this).addClass('product-hover'); },            // 鼠标移入效果增强
        'mouseleave': function () {             // 移出效果消失
            var self = $(this), checked = self.attr('checked');              // 获取选中状态
            if (!checked) self.removeClass('product-hover');              // 未选中时触发
        },
        'click': function () {
            list_items.removeClass('product-hover').removeAttr('checked');             // 移除所有选中效果
            $(this).addClass('product-hover').attr('checked', true);              // 添加当前元素选中
            var index = list_items.index(this);       // 获取商品的位置索引
            if (type == 'cookie') setCookie(cache_key, index);              // Cookie方式存储
            else if (type == 'storage') sessionStorage.setItem(cache_key, index); // sessionStorage存储
        }
    });
    btn_buy.bind('click', function (e) {        // 监听购买按钮单击事件
        e.preventDefault();                     // 阻止默认事件
        var current_select_index, select_item;
        if (type == 'cookie') {
            current_select_index = getCookie(cache_key);
            // Cookie方式获取
        } else if (type == 'storage') {
            current_select_index = sessionStorage.getItem(cache_key);
            // sessionStorage获取
        };
        if (current_select_index != null){
            select_item = list_items.eq(current_select_index);
            select_item.trigger('click');      // 触发选中效果
            if (!e.isTrigger) {                 // 手动操作触发
                alert(select_item.find('a').html() + ',' + select_item.find('strong').html());
            };
        };
    });
    btn_buy.trigger('click');                   // 初始化选中状态
</script></html>
```

10.6.3 代码分析

主页面比较简单，包含了两个iFrame对子页面的调用，本例分析针对于子页面。子页面包含了两种存储和获取方式，分别为Cookie和SessionStorage，方式的选择由URL传入的type参数控制。

代码12行~第24行定义了三个工具方法getQueryString、getCookie、setCookie，分别用于获取URL指定参数的数据、获取指定键值的Cookie数据、设置指定键值的Cookie数据。

代码29行~第49行，监听商品列表元素的mouseenter、mouseleave、click事件。监听前两个事件是为了实现鼠标移入移出标注商品的效果。在商品元素的click监听事件中，会获取选中商品在列表中的位置，通过URL传递的type参数判断进行Cookie或者SessionStorage存储。

代码43行~第58行监听购买按钮的click事件。事件触发后，会先判断URL传递的type参数，获取Cookie或者SessionStorage对应键值数据，该数据值即为之前被选中的商品序号。当商品序号存在时，通过该序号获取对应的商品DOM元素，触发该元素选中效果，并弹出选中商品相关提示信息。



10.7 示例7 存储数据的共享

10.7.1 示例效果

HTML 5本地离线存储给开发人员带来了很大地想象空间，本例将采用LocalStorage实现一个Web版PPT展示器和控制器相结合的功能。示例一共由两张页面组成，使用Chrome浏览器新建两个标签页，分别打开展示页和操作页（“存储数据的共享.html”和“存储数据的共享操作页.html”），运行效果如图10.19所示。

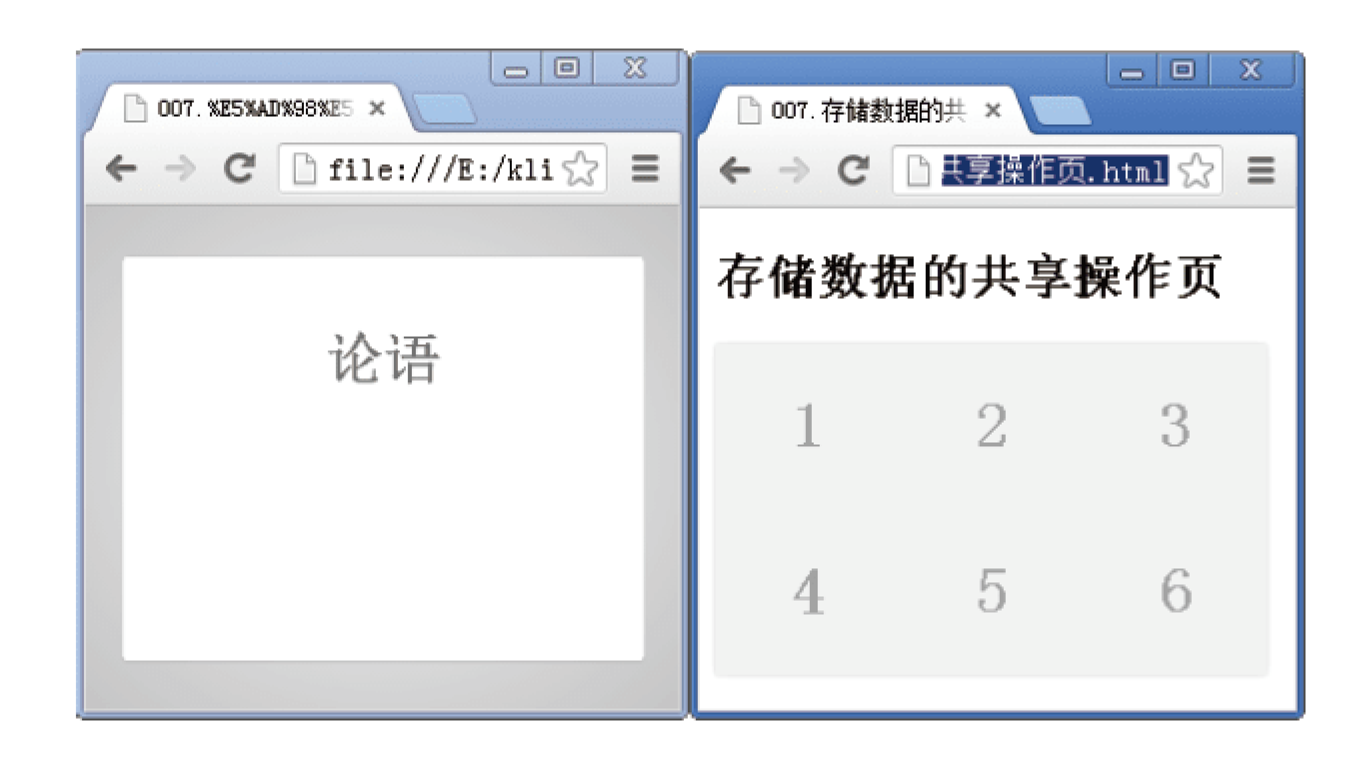


图10.19 使用Chrome打开展示页和操作页

单击操作页面“2”区域，此时展示页切换至第2张幻灯片，效果如图10.20所示。

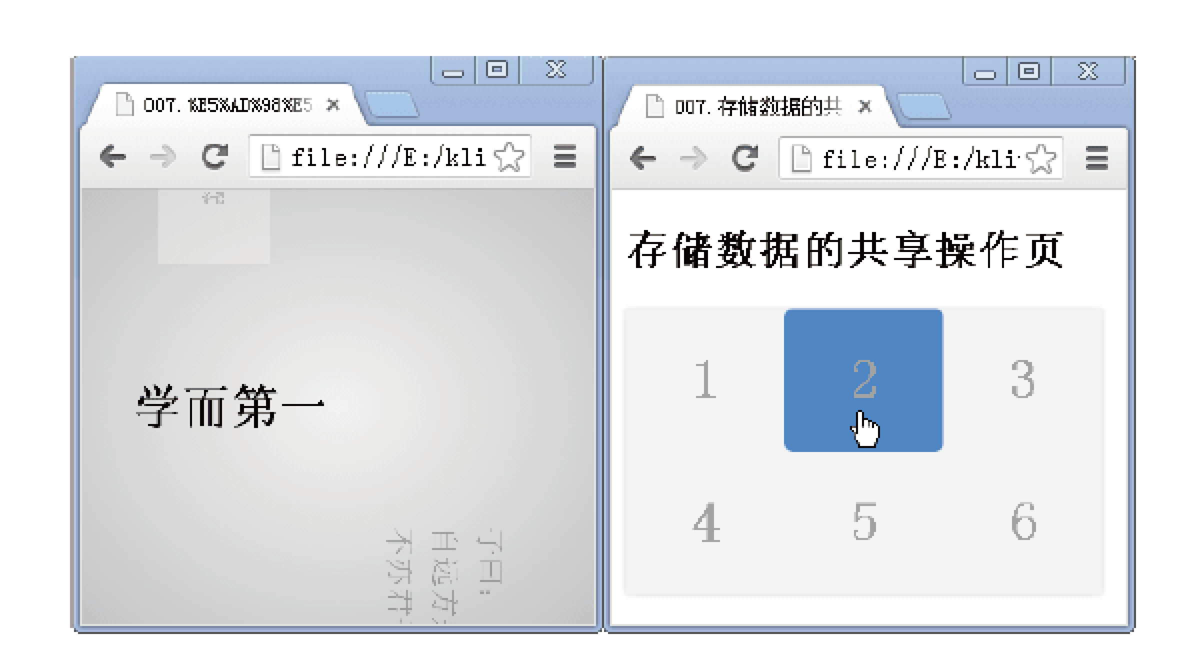


图10.20 单击操作页面“2”

通过前面示例的学习，读者是否已经想明白了该示例的实现方式。本例除了采用LocalStorage实现外，还可以采用传统的Cookie方式实现，但就Cookie和LocalStorage在平常开发技术上的选型问题，笔者认为LocalStorage有两个优势：

- LocalStorage存储空间达到5MB，而Cookie只有4KB。
- Cookie会随每次页面请求被发送至服务器端，消耗带宽，而LocalStorage不会被发送至服务器。



Web版PPT采用GitHub上开源的impress.js项目，项目地址：https://github.com/bartaz/impress.js。

10.7.2 代码设计和分析

本例是一个偏创意型的示例，技术层面的实现非常简单，不过下面还是会做一个简要的分析。首先看PPT主体展示页面，幻灯片HTML结构代码如下：

```
<div id="impress">                                           <!-- 主容器 -->
    <div class="step slide" data-x="-1000" data-y="-1500" >
                <!-- 第一章幻灯片容器 -->
        <q><h1>论语</h1></q>                          <!-- 第一张幻灯片内容 -->
    </div>
    /* ......省略剩余幻灯片，详见本书网络资源 */
</div>
```

impress.js项目默认的主容器ID为impress，其中每张幻灯片需要含有一个step样式类，同时可以在每张幻灯片DIV容器上设置data-x、data-y等自定义属性，表示CSS 3 Transform的相关动画设置。

主页面脚本代码如下：

```
var instance_impress = impress();                       // 获取实例对象
instance_impress.init();                                // 初始化幻灯片
setInterval(function () {
    var step = localStorage.getItem(' slide_step ');      // 获取缓存幻灯片序号
    if (step) instance_impress.goto(parseInt(step));     // 进入第step张幻灯片
}, 100);
```

由于采用原生录入的关系，可以看到“过期时间”一栏并没有信息。单击每条信息后方的“删除”按钮，清空列表内容。清空后，单击“添加”按钮，出现输入浮层并录入测试信息，效果如图10.24所示。



图10.24 单击“添加”按钮

单击“确认”按钮，浮层关闭，列表中出现对应的录入信息，同时“过期时间”一栏出现当前时间加上录入秒数之和的数值，效果如图10.25所示。



图10.25 浮层录入信息

“过期时间”表示该条信息在此时间后过期并消失。10s后单击“刷新”按钮，列表显示为空，过期数据已经消失。

10.8.2 代码设计和分析

完成添加本地离线缓存数据过期功能，外链脚本如下：

```
<script src="../js/jquery-1.8.3.js"></script>
<script src="../js/jquery-ui-1.9.2.custom.js"></script>
<script src="../js/localstorage.js"></script>
<script src="../js/localstorage-expire.js"></script>
```

其中localstorage-expire.js是对localstorage.js中方法的覆盖重写。

下面分析具体的逻辑脚本。首先，初始化相关的元素变量和方法，代码如下：

```
var DB_EXPIRE_NAME = 'html5-storage-local-expire',      // 过期缓存键
    item_tpl = $('#J_item').html(),                      // 列表单个元素模板
```

```
    substitute = function (str, sub) {                  // 字符串格式化函数
        return str.replace(/\{(.+?)\}/g, function ($0, $1)
    { return $1 in sub ? sub[$1] : $0; });
    },
    tbody = $('table.data tbody'),                      // 列表主体
    refresh = $('span.refresh');                        // 刷新按钮
```

监听弹出浮层中确认按钮的单击事件内容，代码如下：

```
var key = wraper.find('input.key').val(),              // 键值
    value = wraper.find('input.value').val(),          // 缓存值
    expire = wraper.find('input.expire').val();        // 过期时间（秒）
if (key.length && value.length && expire.length) {
    var t = new Date();
    t.setSeconds(t.getSeconds() + parseInt(expire));   // 获取计算后的过期时间
    Storage.setItem(key, value, false,          // 缓存值并设置过期时间
                t.getFullYear() + '/' +                // 年
                (t.getMonth() + 1) + '/' +             // 月
                t.getDate() + ' ' +                    // 日
                t.getHours() + ':' +                   // 小时
                t.getMinutes() + ':' +                 // 分钟
                t.getSeconds());                       // 秒
    refresh.trigger('click');                   // 触发“刷新”按钮单击事件
    $(this).dialog("close");                           // 关闭弹出
```

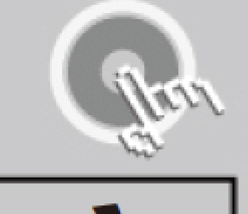
当输入完毕后，将填写的秒数加上当前的时间作为缓存的过期时间，Storage的setItem方法的语法如下：

```
Storage.setItem(key, value, isJSON, expire_date);
```

- key：缓存键。
- value：缓存值。
- isJSON：是否存储为JSON格式。
- expire_date：过期时间。

单击“刷新”按钮，列表中重新展现未过期的LocalStorage数据，代码如下：

```
refresh.on('click', function (e) {
    var length = localStorage.length,   // 获取LocalStorage缓存的数量
        htmls = [],
        expire_data = Storage.getItem(DB_EXPIRE_NAME, true);
        // 获取过期缓存键中的数据
    for (var i = 0; i < length; i++) {
        var key = localStorage.key(i),  // 获取对应索引的缓存键名
           value;
        if (key != DB_EXPIRE_NAME) {    // 判断是否为过期缓存的键名
            value = Storage.getItem(key);                // 获取缓存数据
            if (typeof value != 'undefined' && value !== null) {
                htmls.push(substitute(item_tpl, {
                // 存储行HTML结构至数组
                    key: key,
```



第11章

HTML 5通信大演练

本章将介绍HTML 5带来的两种新通信API，跨文档消息传输和WebSockets技术。其中，WebSockets技术是一个全双工通信信道，开发者可以使用该技术开发实时的、事件驱动的Web应用程序。本章的示例将会带着读者一步步搭建各种场景的通信应用，如跨文档的数据传输、基于WebSockets技术的聊天室等。

本章知识点：

- 跨文档消息传输和WebSockets
- 微博消息实时推送
- 在线代码编辑器
- 通过WebSocket创建聊天室



11.1 示例1 微博消息实时推送

11.1.1 示例效果

HTML 5提供了非常方便的通信机制，利用其跨文档间的通信可以非常容易地实现一些业务功能。本示例演示利用HTML 5的跨文档通信功能，实现微博页面的实时刷新功能。

使用Chrome浏览器打开“微博消息实时推送-index”网页文件，运行效果如图11.1所示。



本示例程序需要通过http://localhost地址运行。

微博消息实时推送

图11.1 页面初始化

过几秒钟后，页面显示效果如图11.2所示。

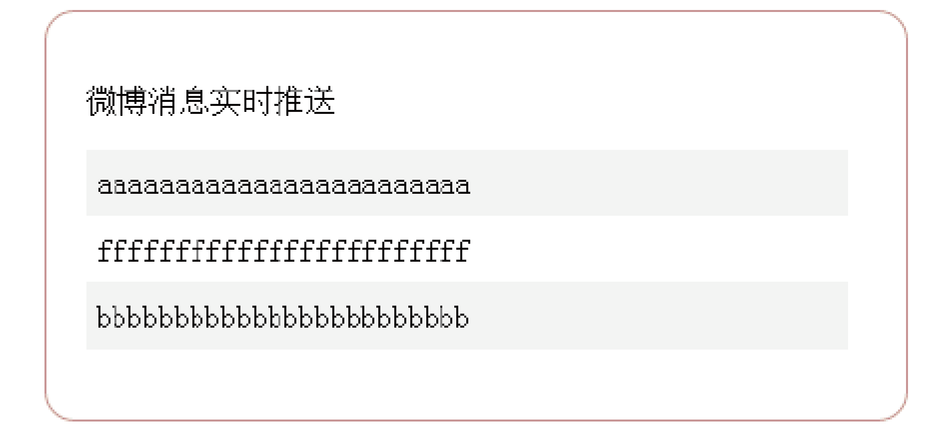


图11.2 微博实时消息



根据运行环境不同，等待的时间会不太一样，这取决于程序的随机计算。

11.1.2 代码设计与分析

利用编辑器打开“微博消息实时推送-index.html”文件，如无特殊说明，以下称此文件为主文件，代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>微博消息实时推送</title>
    <style type="text/css">
    body{
      margin:50px auto;width:400px;padding:20px;border:1px solid #c88e8e;
      border-radius: 15px;                          /* 设置圆角 */
         height: 100%;
    }
    #weibolist{list-style: none; padding: 0;}
    #weibolist li{ background: #F3F3F3;line-height: 24px;height: 24px;padding: 5px; width: 95%;}
    #weibolist li:nth-child(2n){background: #fff;} /* 设置微博列表的偶数项目样式 */
    #weibo-iframe{display: none;}      /* 微博获取iframe设置为不可见 */
    </style>
</head>
<body>
    <p>微博消息实时推送</p>
    <ul id="weibolist"></ul>
    <iframe id="weibo-iframe" src="001.微博消息实时推送-server.html"></iframe>
</body>
<script>
;(function(W){
    var doc = W.document;
    var weibolist = doc.querySelector("#weibolist");
    var handler = function(weibo_data){                // 处理新的微博信息
```

```
28          var li = document.createElement("li");
29          li.innerText = weibo_data;
30          weibolist.appendChild(li);      // 把微博信息显示在weibolist控件中
31      };
32      // 监听从“微博消息实时推送-server.html”页面是否有postMessage请求
33      W.addEventListener('message', function(evt) {    // message事件代理
34          if(evt.origin === 'http://192.168.3.100'){
           // 判断发消息的来源是否正确
35              han+dler(evt.data);         // 把新的微博消息交给处理函数处理
36          }
37      }, false);
38  }(window));
39  </script>
40  </html>
```



这是本示例代码的第一个文件，该文件作为程序的主文件。

第21行，引入微博消息实时处理的Server处理文件，即第二个文件，如无特殊说明，以下称此文件为Server辅助文件。因为要获取数据，需要经常和Server通信，如果把与Server进行通信的工作交给主文件处理，那么主文件的性能损耗较大。通过HTML 5的通信机制，可以通过引入Server辅助文件。当Server辅助文件与Server通信后，获得了新的微博数据，则主动通知主文件有新的微博消息，需要主文件处理。主文件、Server辅助文件、Server三者的关系如图11.3所示。

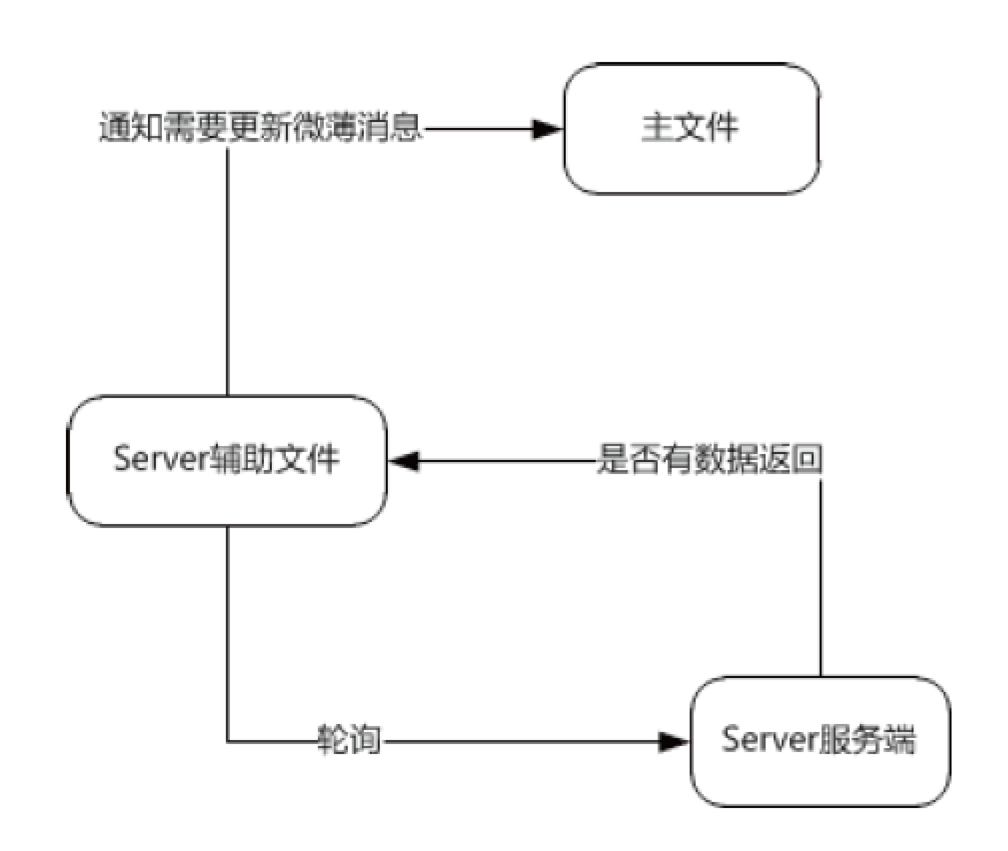


图11.3　主文件、Server辅助文件、Server三者关系

代码第33行，监听Server辅助文件是否发来通知，如果Server辅助文件发出通知，则触发主文件的message事件，然后执行事件的回调函数。



window.addEventListener(‘message’,callback)是HTML 5中新增的事件监听方法，可以接收其他文档通过postMessage发来的消息。有关详细的信息可参考http://dev.w3.org/html5/postmsg/。

第34行，判断消息来源是否可靠，message事件代理中的参数包含两个重要的信息，origin和data。origin保存了消息的来源，data保存了消息的消息体。其他有关信息可参考http://

dev.w3.org/html5/postmsg/。

第35行，如果通过了来源检测，则调用处理函数处理消息。

第27行~第31行，是具体的处理逻辑，读者可根据业务自行扩展。

利用编辑器打开“微博消息实时推送-server.html”Server辅助文件，代码如下：

```
01 <html>
02 <head>
03 </head>
04 <body>
05         <script type="text/javascript">
06                 ;(function(){
07                         // 模拟的数据，真实环境中，数据可以从服务器取得
08                         var virtualData = [
09                                 'aaaaaaaaaaaaaaaaaaaaaaaa',
10                                 'bbbbbbbbbbbbbbbbbbbbbbbb',
11                                 'cccccccccccccccccccccccc',
12                                 'dddddddddddddddddddddddd',
13                                 'eeeeeeeeeeeeeeeeeeeeeeeee',
14                                 'ffffffffffffffffffffffff',
15                                 'gggggggggggggggggggggggg'
16                         ];
17                         /**
18                          * 这里获取是否有新的微博信息产生的逻辑
19                          * 如果有，则返回微博信息
20                          * 如果没有，则返回 null
21                          */
22                         // 从服务器获取数据。此函数为模拟环境，真实环境请从服务器读取
23                         var getData = function(){
24                                 var index = Math.floor(Math.random() * 10);
                                   // 获取模拟数据
25                                 return virtualData[index] || null;
                                   // 如果数据不存在，返回null
26                         }
27                         var aniloop = function(){
28                                 var randTime = Math.floor(Math.random()
    * 10000);
29                                 setTimeout(function(){
30                                                 var data = getData();
                                                           // 判断是否有新微博消息
31                                                 if(data !== null){
                                                           // 判断是否有新微博
32                                                 // 发消息，通知父页面有新消息到了
33                 window.parent.postMessage(data, 'http://localhost');
34                                                 }
35                                                 aniloop();
                        // 在下一阶段继续调用处理程序
36                                         }, randTime);
37                         }
38                         aniloop();
39                 }());
```

```
40         </script>
41 </body>
42 </html>
```

第08行，为了模拟Server返回数据，此数据为伪造。

第24行~第25行，随机获取一个0~9的整数index，并判断virtualData是否存在index的索引。如果存在，返回其值，不存在返回null。

第27行，定义了一个轮询函数，轮询间隔为1~10s。

第32行，如果有微博更新，则使用HTML 5的postMessage向windows.parent（即主文件）发消息。postMessage的调用对象为window.parent，即主文件。也就是说向谁发消息，需要获取对方的引用。该方法包含2个参数，第1个参数为需要发送的内容，一般为文本，有些浏览器也支持对象、数组等。第2个参数为允许的发送源。



postMessage有关详细信息请参考http://dev.w3.org/html5/postmsg/。



11.2 示例2 在线代码编辑器

11.2.1 示例效果

在讲解技术或者做内部分享时，有时候需要一个环境去运行一段JavaScript或者CSS代码等，而且有时候还需要现场修改代码。本示例设计了一个代码实时修改，实时运行的页面，支持边写边运行看效果的功能。

使用Chrome浏览器打开“在线代码编辑器.html”网页文件，运行效果如图11.4所示。



图11.4 代码编辑器初始界面

在左边文本框中输入代码，如下：

```
<html>
	<head>
		<style>
			body{background:#gray;}    <!-- 设置背景色为红色 -->
		</style>
	</head>
	<body>
		<div id="demo">编辑器的代码在这里执行</div>
		<script>
			var body = document.querySelector("body");  // 获取body
				setTimeout(function(){  // 设置2s后背景变成蓝色
					document.querySelector("#demo").innerHTML = "2s后执行到这里";
				}, 2000);          // 设置定时2s执行回调函数
			alert("边写边练习, 2s后背景变蓝色"); // 测试代码是否运行
		</script>
	</body>
</html>
```

然后单击“运行”按钮，运行效果如图11.5所示。



图11.5 弹出提示语

在弹出框中，单击“确定”按钮，过2s后运行效果如图11.6所示。

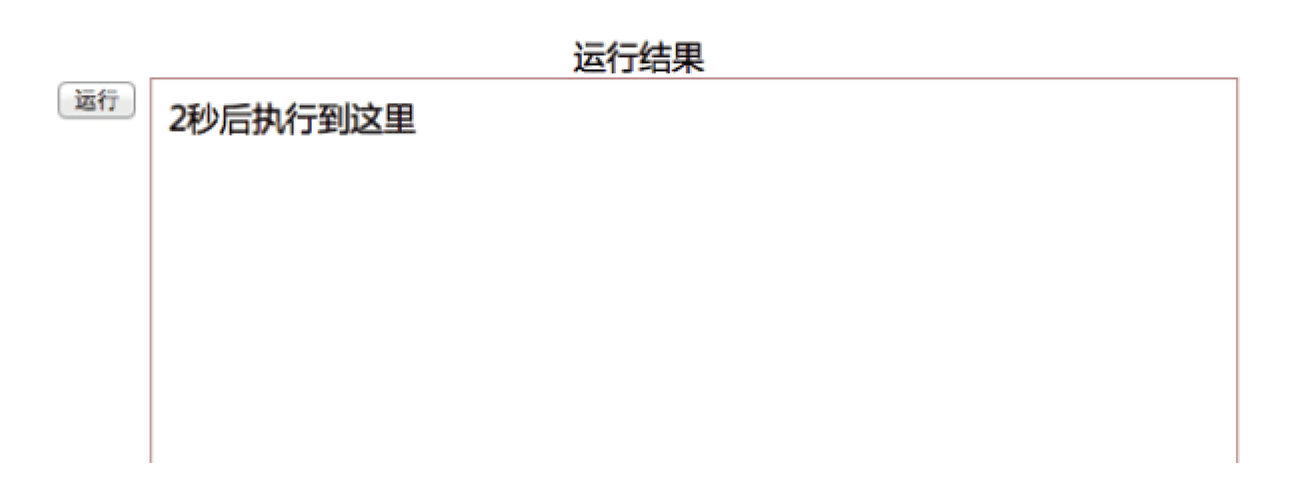


图11.6 运行最终结果

11.2.2 代码设计与分析

利用编辑器打开“在线代码编辑器.html”文件，如无特殊说明，以下称此文件为代码文件，代码如下：

```
46 </script>
47 </html>
```

第37行~第44行，获取代码文件框<textarea id=” code_writing” >的值，把这个值发送到windows.frames[0]文档中，即发送到“实现代码演示示例-runing.html”文档中。

第43行，使用postMessage发送数据，使用方法请参考“微博消息实时推送.html”。

利用编辑器打开“在线代码编辑器-结果.html”文件，如无特殊说明，以下称此文件为结果文件，代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="utf-8">
      <title>在线代码编辑器-结果</title>
</head>
<body>
</body>
<script>
 var this_html = document.querySelector("html");
      // 监听“在线代码编辑器.html”页面是否有postMessage请求
      window.addEventListener('message', function(evt) {
      // message事件代理
             if(evt.origin === 'http://localhost:8080'){
             // 判断发消息的来源是否正确
                    var html = document.createElement("html");
                    // 创建<html>标签
                    var data = "" + evt.data;
                    // 将evt.data转成字符串
                    html.innerHTML = evt.data;
                    // 设置<html>标签的内容
                    var _reg=/<script[^>]*>.*(?=<\/script>)<\/script>/gi;
                    // 匹配<script>标签的正则表达式
                    data = data.replace(_reg,"");
                    // 替换<script>标签为空
                    var _scripts = html.getElementsByTagName("script");
                    // 获取传过来的所有<script>
                    this_html.innerHTML = data;
                    // 替换当前页面的HTML的内容
                    var body = this_html.querySelector("body");
                    // 获取当前页面的body标签
                    if(_scripts.length){       // 判断是否有JavaScript语句
                           for(var i = 0, length = _scripts.length; i < length; i++){
                                  var script = document.createElement("script");
                                  if(_scripts[i].src){
                                  // JavaScrip是外部引用文件
                                         script.src = _scripts[i].src;
                                         // 则设置其src属性
                                  }else{    // 否则设置其为内联JavaScript
                                  script.innerHTML = _scripts[i].innerHTML;
                                  }
                                  body.insertBefore(script);
```

```
                                                 // 把<script>标签插入到body前执行
31                               }
32                        }
33                 }
34          }, false);
35   </script>
36   </html>
```

第12行，设置message事件代理，当用户单击代码文件中的运行按钮时，代码文件页面的postMessage函数将把当前用户写的代码发送到结果文件中。结果文件通过监听message事件，可以获得这份代码。

第13行，为了安全起见，需要判断信息来源是否正确。

第14行~第32行，message事件的处理程序。因为传过来的代码是字符串类型，所以可以利用innerHTML属性将其转化为元素对象，见第14行和第16行代码。

第19行，获取代码里的script标签。如果直接把script标签放入到文档中，其JavaScript代码不会执行，但可以通过getElementsByTagName函数获取到所有的script标签。

第20行，删除结果文件的HTML，并替换成用户写的HTML。



并不会删除结果文件中已经存在的JavaScript，也就是说，即使HTML被完全替换了，但当前的onmessage事件代理仍然是有效的。当用户再次在代码文件中单击运行按钮时，该事件处理程序仍然会被执行。

第24行~第29行，动态创建script标签，如果script标签包含src属性，则视其为外部引用，否则视为内联JavaScript。

第30行，把script标签插入到body之前，然后JavaScript代码开始执行。



11.3 示例3 在iFrame中嵌入可变的编辑器

11.3.1 示例效果

在做编辑器开发中，一般是把编辑器放入到一个iFrame中，如果在iFrame中改变编辑器的大小，iFrame的大小需要随之改变。本示例演示了一个写博客文章的编辑器。把写文章的逻辑放到一个iFrame中，当iFrame中的编辑器尺寸改变时，iFrame也随之改变。

使用Chrome浏览器打开“在iframe中嵌入一个可变大小的编辑器.html”网页文件，运行效果如图11.7所示。

图11.7 将发布文章逻辑嵌入到iFrame中

单击“700×400”选项按钮，运行效果如图11.8所示。

图11.8 改变编辑器尺寸

11.3.2 代码设计与分析

利用编辑器打开“在iframe中嵌入一个可变大小的编辑器.html”文件，如无特殊说明，以下称此文件为主文件，代码如下：

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>在iframe中嵌入一个可变大小的编辑器</title>
<script src="../js/jquery-1.8.3.js"></script>
<style>
      .invisible{display: none;}
      .iframe{
            width:400px;
            height:390px;
            border:1px solid #ccc;
            border-radius: 5px;             /* iframe 边框圆角 */
            -webkit-box-shadow:5px 5px rgba(0,0,0,.6);
      /* iframe 阴影 */
            box-shadow:5px 5px rgba(0,0,0,.6);      /* iframe 阴影 */
      }
</style>
</head>
<body>
<p>这里是页面上的其他内容</p>                    <!--页面上正常的内容-->
<textarea class="invisible">                    <!--设置文本框架不可见-->
<!--iframe放入到 textarea中-->
      <iframe scrolling="no" class="iframe" frameborder="0" src="003.在
      iframe中嵌入一个可变大小的编辑器内容页.html"></iframe>
</textarea>
</body>
<script>
      $(function(){
            var invisibles = $(".invisible");        // 获取textarea节点
            var iframe = parseDom(invisibles.text());
            // 将textarea中的iframe转化DOM
            invisibles.replaceWith(iframe);   // 把textarea替换成iframe
            iframe = $("iframe");             // 获取页面上的iframe
            window.addEventListener('message', function(evt){
            // 监听message事件
                  var iframeOffset = evt.data || {};
                  // 获取监听到的数据
                  iframe.css({                // 重新设置iframe的宽与高
                        width: iframeOffset.width + 50,
                        // 宽度偏移50px
                        height: iframeOffset.height + 190,
                        // 高度偏移190px
                  });
            });
      });
      function parseDom(arg) {                // 将字符串转成DOM函数
            var objE = document.createElement("div");
            objE.innerHTML = arg;
            return objE.childNodes;
      };
</script>
</html>
```

```
35                </div>
36                </form>
37          </div>
38  </body>
39  <script>
40          var txt = $(".content");                       // 获取文章内容控件
41          $(".slt").bind("click", function(){            // 绑定文本框尺寸的单击事件
42                  var cat = this.dataset.cat;            // 获取当前所选择的尺寸序号
43                  var offset = {                         // 计算当前的文本框大小
44                          'width': cat * 350,
45                          'height': cat * 200
46                  };
47                  $(".slt").removeClass("selected")       // 取消所有尺寸选择的选中状态
48                  $(this).addClass("selected");
                            // 把当前选择的尺寸设为选中状态
49                  txt.css(offset);                               // 设置文本框的新尺寸
50                  window.parent.postMessage(offset, '*');
                    // 把当前新的尺寸通知给主文件页
51  });
52  </script>
53  </html>
```

第14行，在本示例中，因为不处理后端数据，所以在onsubmit事件中使用return false处理，防止表单提交。

第24行~第26行，是三种编辑框的尺寸，供用户选择，分别是350×200、700×400、1050×600。

第30行，文本编辑器控件。用户选择编辑器尺寸后，该控件的大小会随着变化。

第42行，获取用户选择的尺寸类型，其包含在dataset的cat属性值中。

第43行~第46行，计算当前选择的编辑器的实际尺寸，宽度为当前cat值与350乘积，高度为当前cat值与200乘积。

第47行~第49行，用户选择尺寸类型后，先设置当前选中状态，在把当前编辑器的宽与高替换成当前选择的结果。

第50行，因为该页面被嵌入在一个iFrame中，所以当编辑框的尺寸改变后，iFrame的宽与高还是固定的，导致编辑器显示不全或者外边框过大，该行代码把当前用户选择的结果通过postMessage发送给主文件进行iFrame宽与高的重新设置。



11.4 示例4　预览网站内容

11.4.1 示例效果

在新闻阅读类网站中，经常需要处理的一个场景是把新闻信息嵌入到一个iFrame中供用户阅读详细的新闻。而这种情况一般的处理流程是跳到一个新的页面，本示例通过HTML 5的postMessage通信，可以不用跳转，而是直接在当前页面新开一个页面，供用户阅读。

使用Chrome浏览器打开“预览网站内容.html”网页文件，运行效果如图11.9所示。

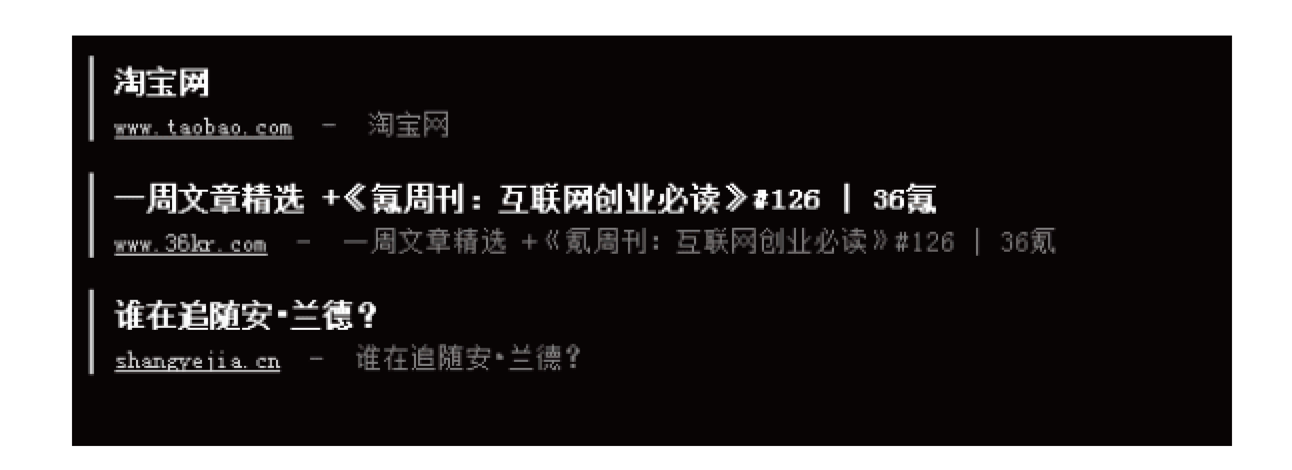


图11.9 新闻列表

单击淘宝网标题链接，运行效果如图11.10所示。



图11.10 预览网站具体内容

单击左侧“关闭”按钮，如图11.11所示，页面回到图11.9所示状态。



图11.11 单击“关闭”按钮

11.4.2 代码设计与分析

利用编辑器打开“预览网站内容.html”文件，如无特殊说明，以下称此文件为主页面，代码如下：

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
      <meta charset="utf-8">
      <title>预览网站内容</title>
      <script type="text/javascript" src="../js/jquery-1.8.3.js"></script>
      <style type="text/css">
      // 样式内容见代码源文件
      </style>
</head>
<body>
      <div class="news">
             <!--新闻列表-->
             <div class="clist clearfix">
                    <div class="clist-center clist-noavatar">
                           <h3><a href="http://taobao.com/" class="item" target="_blank">淘宝网</a></h3>
                           <div class="clist-con">
                                  <a href="http://taobao.com/" class="item" target="_blank">www.taobao.com</a>
                                         <span>  - </span>  淘宝网
                           </div>
                    </div>
             </div>
             <!--更多阅读列表见代码源文件 -->
             <!--// 新闻列表 end-->
      </div>
      <iframe src="004.preview.html" id="iframe"></iframe><!--预览页面 -->
</body>
<script>
$(function(){
      var body = document.querySelector("body"),     // 获取body控件
             iframe = document.querySelector("#iframe"),
             // 获取iframe控件
             win = iframe.contentWindow;      // 获取iframe的document
      $(".item").on("click", function(e){      // 绑定每一个新闻标题的单击事件
             e.stopPropagation();             // 阻止事件冒泡
             e.preventDefault();              // 阻止事件的默认行为
             // var style = window.getComputedStyle(iframe);
             // 获取iframe的样式对象
             // 设置样式
             body.style.height = document.documentElement.clientHeight;
             iframe.style.display = 'block';
             iframe.style.height = (document.documentElement.clientHeight - 15) + 'px';
             var width = (document.documentElement.clientWidth - 50) + 'px';
             win.postMessage({                // 发送消息给iframe
```

```
    ">收藏</a>
13              <a href="javascript:;" onclick="alert('请自己扩展此功能');
    ">评论</a>
14              <a href="javascript:;" id="close">关闭</a>
15          </p>
16          <iframe src="" id="iframe"></iframe>    <!—加载第三方的内容>
17  </body>
18  <script>
19  ;(function(W){
20      var doc = W.document;
21      var parent = window.parent;
22      var iframe = document.querySelector("#iframe");
23      var close = document.querySelector("#close");
24        window.addEventListener('message', function(evt) {
          // 监听message事件
25              iframe.src = evt.data.href;            // 设置iframe的src
26              iframe.style.height = evt.data.height; // 设置iframe的高度
27              iframe.style.width = evt.data.width;   // 设置iframe的宽度
28              // console.log(evt.data);
29        }, false);
30        close.addEventListener("click", function(){
31              window.parent.postMessage("closed", "*");
                // 发送“关闭详情页面”动作
32        }, false);
33        // 当文档加载完毕，给父级来源发送信息。
34        window.addEventListener('load', function(e){
35              window.parent.postMessage("ready", "*");
                // 告诉主页面“已经准备好”，可以向我发消息
36        }, false);
37  }(window));
38  </script>
39  </html>
```

第13行~第15行，详情页面的动作控件页面，本示例模拟了关闭动作。

第24行~第29行，监听从主页面发来的信息。一旦收到主页面发来的信息，立即执行，并设置iFrame的src属性、宽度和高度。

第30行，监听动作面板的关闭事件，并通知主页面做关闭操作。



11.5 示例5 定时给客户发消息

11.5.1 示例效果

本示例模拟微博系统中不刷新页面的新消息提醒。一般的微博采用Ajax技术进行HTTP轮询判断是否有新的微博消息。本示例采用EventSource技术，建立5个长连接，如果有新消息

来，则对用户进行提醒。



运行本示例必须在客户端安装Node.js，Node.js是一个跨平台的基于V8引擎的JavaScript编译器，具体可参考http://nodejs.org。

在本地创建数据库html5，创建表名为“demo_12_5”，数据库脚本如下：

```
CREATE DATABASE html5;
USER html5;
CREATE TABLE `demo_12_5` (
  `id` int(11) NOT NULL AUTO_INCREMENT,                    // 索引序号
  `msg` text NOT NULL,                                     // 消息
  `addDate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,  // 添加时间
  `isRead` tinyint(1) DEFAULT '0',                         // 是否已读
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8;
```

在命令行模式下，使用Node命令运行“005.server.js”文件，命令行如下：

```
node 005.server.js
```

运行效果如图11.12所示。



图11.12 启动Node.js服务



如果运行失败，有可能是机器的端口被占用，请在网络资源中更改端口号。

使用Chrome浏览器打开http://localhost:8000地址，运行效果如图11.13所示。

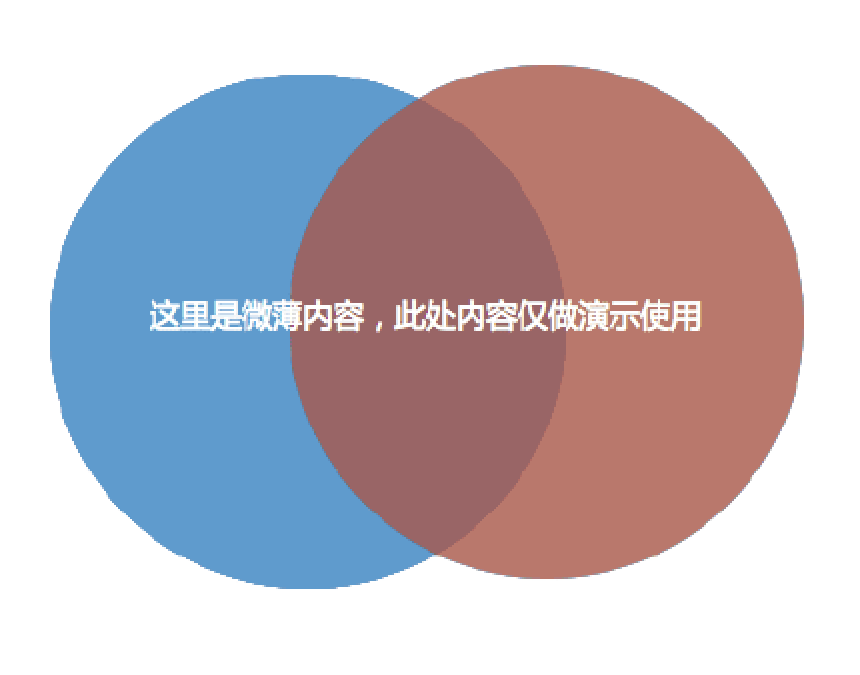


图11.13 无未读消息时的状态

单击进入后台添加消息按钮，Chrome浏览器新打开一个标签页，运行效果如图11.14所示。

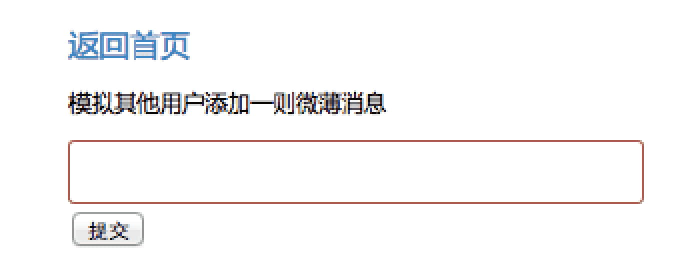


图11.14 模拟添加微博消息界面

在文本框中输入一则消息“测试消息”，单击“提交”按钮。运行效果如图11.15所示。

添加成功， 返回首页 **或者** 继续添加

图11.15 模拟微博消息成功

在Chrome浏览器返回“定时给客户发消息”标签页面，可以看到页面顶部出现了“有1条新消息”的提示，如图11.16所示。

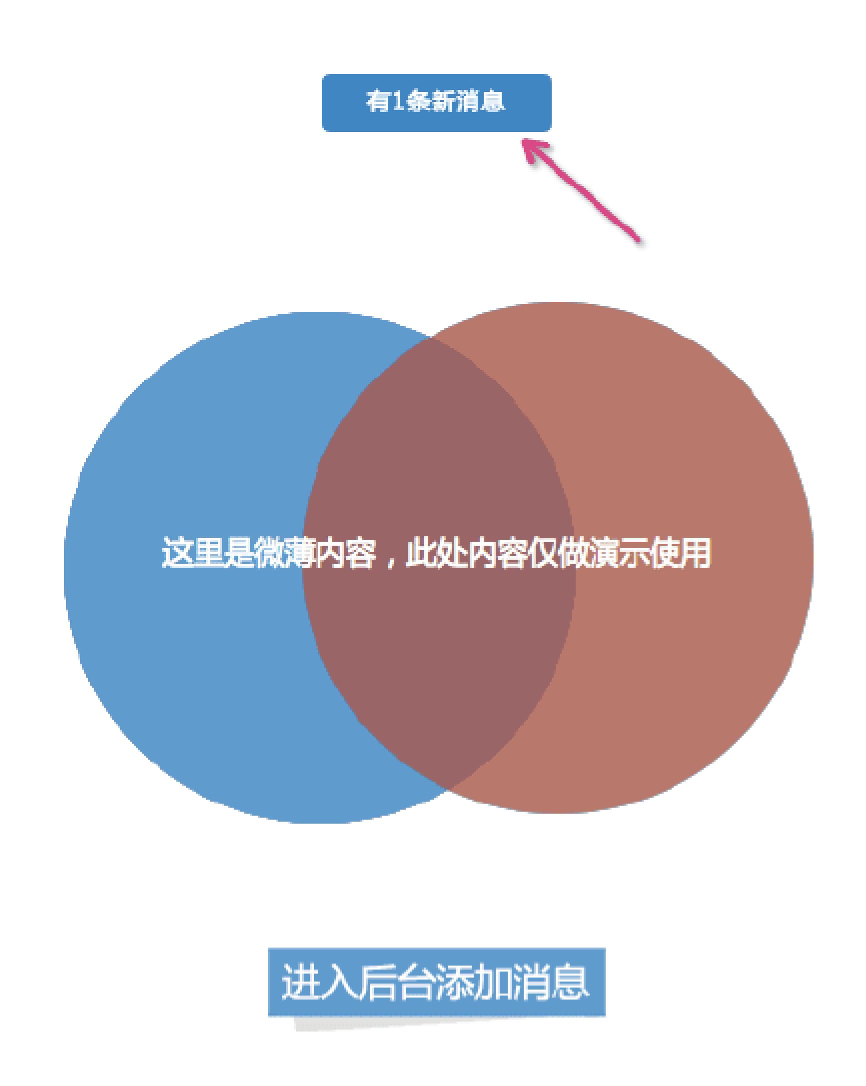


图11.16 后台检测到有新消息，并将其推送到前台提醒



读者可以继续到“模拟添加微博消息后台”页面，添加更多的消息，在切换回“定时给客户发消息”页面，查看是否有新的微博消息提醒。

11.5.2 代码设计与分析

利用编辑器打开“server.js”文件，如无特殊说明，以下称此文件为Server文件，代码如下：

```
var config = {                                          // 配置文件
    db: {
            host   : 'localhost',                       // 数据库连接地址
            user   : 'root',                            // 数据库用户名
            password: 'root',                           // 数据库密码
```

```
            database: 'html5'                           // 数据库名
        },
        port: 8000                                      // 服务端口号
};
var http = require('http');                             // HTTP模块
var sys = require('sys');                               // 系统模块
var fs = require('fs');                                 // 文件系统模块
var mysql = require('mysql');                           // 数据库连接模块
var url = require('url');                               // URL解析模块
var connection = mysql.createConnection(config.db);
// 创建数据库连接
connection.connect();                                   // 连接数据库
http.createServer(function(req, res) {                  // 创建http server
        if (req.headers.accept && req.headers.accept ==
 'text/event-stream') {
                if (req.url == '/notice') {             // 判断请求的URL
                        res.writeHead(200, {
                                'Content-Type': 'text/event-stream',
                                'Cache-Control': 'no-cache',
                                'Connection': 'keep-alive'
                        });
                        loop(req, res);
                } else {
                        console.log('404');
                        res.writeHead(404);
                        res.end();
                }
        } else if(req.url == '/admin'){                 // 后台模拟添加消息
                res.writeHead(200, {'Content-Type': 'text/html'});
                res.write(fs.readFileSync(__dirname + '/005.admin.html'));
                // 载入admin.html文件
                res.end();
        } else if(req.url.indexOf('/addmsg') !== -1 ){      // 添加消息处理
                var url_parts = url.parse(req.url, true);   // URL解析
                var query = url_parts.query;                // 获取参数
                res.writeHead(200, {'Content-Type': 'text/html'});
                if(query.msg){
                        var msg = {msg: query.msg};         // 获取消息内容
                        // 插入数据到数据库中
                        connection.query("insert into demo_12_5 set ?",
 msg, function(err, result){
                                console.log(result);
                                res.write("添加成功, <a href='/'>返回首页</a>
或者 <a href='/admin'>继续添加</a>");
                                res.end();
                        });
                }else{                                      // 添加失败逻辑
                        res.write("添加失败, <a href='/admin'>重新添加</a>");
                        res.end();
                }
        } else {                                            // 默认页面(首页)
```

```
52              res.writeHead(200, {'Content-Type': 'text/html'});
53              res.write(fs.readFileSync(__dirname + '/005.定时给客户发
    消息.html'));
54              res.end();
55          }
56  }).listen(config.port);
57  console.log('listen on http://localhost:' + config.port);
                // 在控制台输出端口号
58  function loop(req, res){                              // 后台轮询函数
59          getMoreSMS(res, function(data){               // 查看是否有新消息
60              sendSSE(req, res, data);                  // 通知客户端有新消息
61          });
62          //在一个长链接中每隔10s服务端进行事件推送到客户端
63          setInterval(function() {
64              getMoreSMS(res, function(data){
65                  sendSSE(req, res, data);
66              });
67          }, 10000);
68  }
69  function sendSSE(req, res, data) {
70          var id = (new Date()).toLocaleTimeString();   // 获取当前的时间戳
71          res.write('id: ' + id + '\n');                // 发送时间戳
72          res.write("data: " + JSON.stringify(data) + '\n\n');
            // 发送数据（JSON格式化）
73  }
74  function getMoreSMS(res, callback){
75          // 查询数据库，查找所有未读的消息
76  connection.query('SELECT id, msg, addDate from demo_12_5 where isRead
    = 0', function(err, rows,fields) {
77              if (err || !rows.length) {
78                  console.log('errors');
79              }else{
80                  callback(rows);
81              }
82          });
83  }
```

代码第01行~第09行，为服务器的配置文件，配置数据库的信息和服务启动的端口号，第03行~第06行分别是数据库连接地址、数据库用户名、数据库密码、数据库名。第08行为服务启动的端口号。



如果服务启动时，提示端口号被占用，需要在第08行修改成一个未被使用的端口号。

第10行~第14行，包含本示例需要用到的Node.js模块。

第15行~第16行，创建一个数据库连接实例，数据库连接类库采用的是node-mysql。



更多关于node-mysql的信息请参考GitHub开源项目https://github.com/felixge/node-mysql。

```
21              var data = JSON.parse(e.data),    // 解析服务器推送过来的数据
22                     newMsgNum = data.length;   // 计算新消息数据
23                     msg.style.display = 'block';      // 显示消息区域
24                     msg.innerHTML = "有" + newMsgNum + "条新消息";
                       // 打印消息数量
25        }, false);
26        source.addEventListener('open', function(e) {  // 监听事件源打开时
27              console.log('open');
28        }, false);
29        source.addEventListener('error', function(e) { // 监听事件源错误时
30              console.log('error');
31          if (e.readyState == EventSource.CLOSED) {   // 如果事件源连接被关闭
32              console.log('Connection was closed'); // 打印出事件已经被关闭
33          }
34        }, false);
35  }else{
36        alert('您的浏览器out了！');                 // 浏览器不支持EventSource
37  }
38  </script>
39  </html>
```

代码第08行为新消息提醒区域，如果消息数小于0，默认不显示，当未读消息数大于0时显示未读消息数。

第10行，微博信息列表区域。在该示例中，微博消息列表不是本示例的核心，所以采用区域代替，读者可根据实际情况填充。

第11行，后台添加模拟消息的入口，因为是模拟消息列表，所以该入口在实际环境中并不存在。

第14行，判断浏览器是否支持EventSource技术。

第15行，实例化EventSource。参数设置为“/notice”，是告诉服务器的请求地址为“/notice”，对应“server文件”的第19行程序逻辑。

第17行~第19行，监听用户单击未读消息数的按钮，该示例并未扩展如何清除未读消息，请读者结合实际环境进行清除。

第20行~第25行，监听服务器端发来的消息，如果服务器端发来新消息，表明有新的未读消息，程序重新统计未读消息数。第24行，把新的消息数显示在新消息提醒区域上。

第26行，监听事件打开状态。当服务端与客户端建立起连接时，可以通过open事件做一些逻辑处理。

第29行，监听事件的错误状态。当服务端与客户端建立连接发生错误时，该事件被调用。

利用编辑器打开“admin.html”文件，如无特殊说明，以下称此文件为后台模拟消息文件，代码如下：

```
<!DOCTYPE html>
<html>
<title>定时给客户发消息</title>
<style>
      // 样式见代码源文件
</style>
<body>
```

```
08        <div class="admin">
09        <a href="/" class="back">返回首页</a>
10        <p>模拟其他用户添加一则微博消息</p>
11        <form id="msgform" method="GET" action="/addmsg">
    <!--添加消息表单-->
12               <input type="text" id="msg" name="msg" />
    <!--添加消息文本输入框-->
13               <br />
14               <input type="submit" value="提交" />     <!--提交按钮-->
15        </form>
16        </div>
17  </body>
18  </html>
```

第11行~第15行，定义了添加模拟消息的表单，该表单提交到“/addmsg”请求上，对应“server文件”的第35行执行程序。



11.6 示例6 通过WebSocket创建聊天室

11.6.1 示例效果

本示例开发一个聊天系统，客户端采用HTML 5的WebSocket技术，服务端采用Node.js搭建一个支持WebSocket功能的服务器。



运行本示例必须在客户端安装Node.js，Node.js是一个跨平台的基于V8引擎的JavaScript编译器，具体可参考http://nodejs.org。

运行本示例程序，需要先安装Node.js依赖的WebSocket包，在命令行模式下进入本示例的目录，然后运行如下命令：

```
npm install
```

在命令行模式下，使用Node.js命令运行“app.js”文件，命令如下：

```
node app.js
```

运行效果如图11.17所示。

```
130 [redacted] 8 (git)-[master] %sudo node app.js
Password:
服务器运行在：http://127.0.0.1:9090，请用浏览器打开！
```

图11.17 启动Node.js服务

欢迎进入通过Socket创建的聊天系统

通过Socket创建聊天室

欢迎 B，当前时间是 Sun Jul 21 2013 21:17:33 GMT+0800 (CST)

欢迎进入聊天室，目前有2人在聊天

广播: 【B】^-^加入了聊天室 [21:17:33]

发送给 所有人

用户列表

A

B

图11.20 用户B进入聊天室

在Chrome浏览器中切换标签到用户A的聊天室，此时用户A的聊天室可以看到用户B加入到聊天系统中，如图11.21所示。

欢迎进入通过Socket创建的聊天系统



图11.21 用户B加入后用户A的聊天界面

在Chrome浏览器，再新开一个标签页面，重复以上步骤，让用户C也加入到聊天系统中，用户C的聊天室如图11.22所示。

欢迎进入通过Socket创建的聊天系统

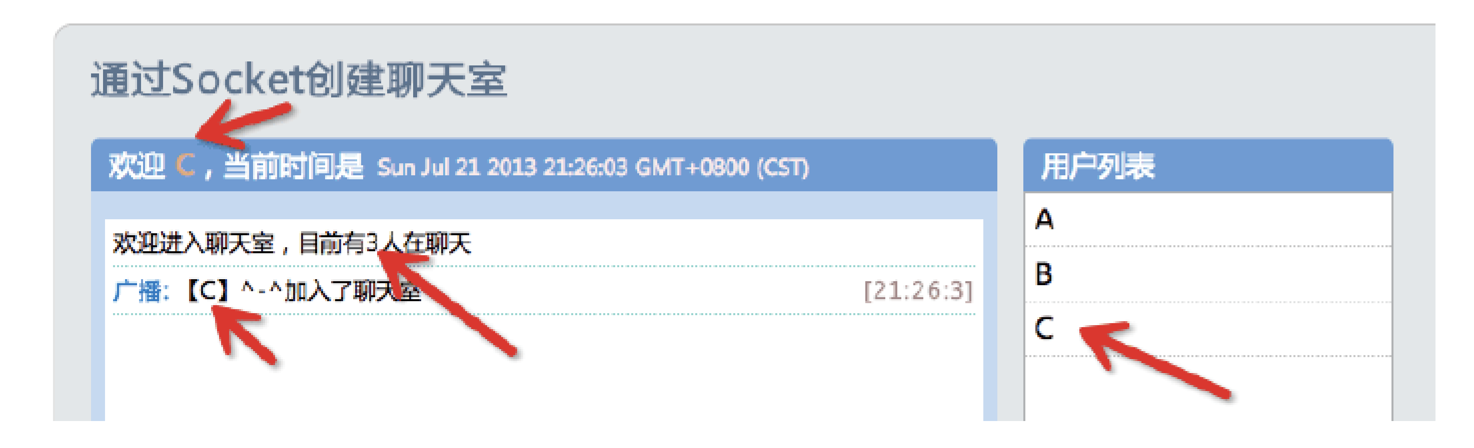


图11.22 用户C的聊天室

此时聊天系统中可以看到有三个用户，分别是用户A、用户B和用户C。当一个用户加入到聊天系统中，系统向所有的用户广播一条消息，通知所有用户“某某”加入到了聊天室里，并更新在线聊天人数和在线用户列表。每一个用户可以向所有人发起聊天，或者对某一个人发起聊天。

在用户A的聊天室里，向所有人发起消息“大家好，我是A”，操作步骤如下：切换到用户A的聊天室，在聊天文本框中输入“大家好，我是A”，然后按Enter键，如图11.23所示。



图11.23　用户A向所有人发消息

在用户A、用户B、用户C的聊天室里都可以看到用户A发的这条消息，如图11.24~图11.26所示。



图11.24　用户A收到用户A发的消息



图11.25　用户B收到用户A发的消息



图11.26 用户C收到用户A发的消息

用户B和用户C看到用户A的消息后，用户B向用户C询问用户A是谁，操作过程如下：切换到用户B的聊天室，在聊天文本框中输入“C，你知道A是谁吗？”，如图11.27所示。



图11.27 用户B向用户C询问

此时用户A无法收到用户B的消息，用户C收到了用户B的消息，用户B和用户C的界面如图11.28和图11.29所示。



图11.28 用户B向用户C发私信



图11.29 用户C收到了用户B的私信

11.6.2 代码设计与分析

利用编辑器打开“app.js”文件，代码如下：

```
44 server.listen(PORT);                    // 静态服务器监听在用户定义的端口
45 console.log("服务器运行在: http://127.0.0.1:" + PORT + ", 请用浏览器打开! ");
46 websocketserver(server);                // 运行web wocket server
```

app.js文件是服务器启动的入口文件，主要包括启动静态服务器和启动WebSocket服务器。

第01行~第08行，加载本系统需要的模块，http、path、url、fs是系统模块，主要用来启动静态服务。config模块包含了一些基本配置信息，在本例中主要需要配置端口号。websocketserver模块，是本示例的核心模块，主要提供WebSocket的请求处理服务。

第10行，解析URL的路径，并对URL做出响应。

第11行，调用系统path模块，判断文件是否存在。

第13行~第16，如果文件不存在则抛出404错误。

第21行~第22行，文件存在，但读取失败，抛出500错误。

第24行~第29行，提取文件后缀，并生成对应的mime。其映射关系是根据config.js文件中的mime做映射。代码如下：

```
"css": "text/css",                              // 样式表
"gif": "imge/gif",                              // gif格式图片
"html": "text/html",                            // html页面
"ico": "image/x-icon",                          // icon格式图标
"jpeg": "image/jpeg",                           // jpeg格式图片
"jpg": "image/jpeg",                            // jpg格式图片
"js": "text/javascript",                        // JavaScript脚本
"json": "application/json",                     // json格式文件
"pdf": "application/pdf",                       // pdf格式文件
"png": "image/png",                             // png格式图片
"svg": "image/svg+xml",                         // svg格式矢量文件
"swf": "application/x-shockwave-flash",         // swf格式flash文件
"tiff": "image/tiff",                           // tiff图片文件
"txt": "text/plain",                            // txt文本文档
"wav": "audio/x-wav",                           // wav音频文件
"wma": "audio/x-ms-wma",                        // wma音频文件
"wmv": "video/x-ms-wmv",                        // wmv视频文件
"xml": "text/xml"                               // xml文件
```

第36行，创建静态服务器。

第38行，设置默认首页为page.html页面。

第40行，启动静态服务器。

第44行，调用WebSocketServer，并把httpServer的实例作为参数传递给WebSocketServer。

利用编辑器打开websocketserver.js文件，代码如下：

```
var
clients = [],                                   // 存储客户端连接
users = [],                                     // 存储连接的用户名
WebSocketServer = require('websocket').server;
// 引入WebSocketServer模块
var run = function(httpServer){
```

```
    wsServer = new WebSocketServer({         // 创建WebSocketServer实例
            httpServer: httpServer,
            autoAcceptConnections: false
    });
    wsServer.on('request', function(request) {
            var connection = request.accept('chat-protocol',
 request.origin)    ;     // 获取连接
            var client_index = clients.push(connection) - 1;
                    // 聊天室人员加1
            users[client_index] = 'Anonymous';      // 设置用户名为
            // Anonymous
            // 当前用户接收客户端发来的消息
            var message = function(message) {         // 接收消息回调函数
                    if (message.type === 'utf8') {
                            var data = JSON.parse(message.utf8Data);
                            var message = data['message'];    // 消息
                            var sender = data['sender'];      // 发送者
                        if(sender == 'ws:__join__'){ // 有新用户加入聊天室
                        // 新加入逻辑
                                    // 广播
                                    broadcasted( '【' + message + '】^-^
加入了聊天室');        // 通知所有人有人加入
                                    // 更新用户名单
                                    users[client_index] = message;
                                    // 更新当前加入者用户名
                                    updateUser();        // 更新用户列表
                            }else if(sender == 'ws:__send__'){
                            // 发送消息逻辑
                                    var from = message.from,    // 获取发送源
                                            to = message.to, // 获取目的地源
                                            msg = message.msg;  // 获取消息体
                                    if(to == 'all'){ // 发送给所有人
                                    // 发送对象是所有用户
                                            broadcasted(msg, from);
                                            // 广播消息
                                    }else{               // 发送给某个人
                                            for (var i = 0; i < clients.
length; i++){
                                                    if(users[i] ==
 to || users[i] == from){          // 发送给自己和对方
                                                            var json_data =
 JSON.stringify({'message': '私信: ' + msg, 'sender':      from});
                                                            clients[i].
sendUTF(json_data);        // 把消息push到客户端
                                                    }
                                            }
                                    }
                            }
                    }
            };
            // 当前加入者监听退出聊天室事件
```

第28行~第30行，获取客户端发送来的发送者身份、发送的消息体和发送的对象。

第29行~第40行，如果发送对象为所有用户，则把当前消息广播给所有人，否则，把这条消息发送给他自己和发送对象，保证只有自己和发送的对象才能看到这条消息。

第45行~第51行，当连接关闭时，调用此回调函数。表示用户离开了聊天室，服务端收到此消息后，把当前离开的客户端删除，然后从当前的用户列表中删除离开的用户，最后通知所有其他用户“某某用户已经离开了聊天室”。

第52行，监听消息接收事件。

第53行，监听连接关闭事件。

第57行~第60行，定义广播消息函数，此函数向所有用户发送消息。clients存储了所有用户的连接请求，利用连接对象connection的sendUTF方法，可以向客户端发送消息。



更多详情请参考https://github.com/Worlize/WebSocket-Node/wiki/Documentation。

第62行，向所有客户端更新用户列表。

利用编辑器打开“www/page.html”文件，代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="content-type" content=
"text/html;charset=utf-8">
        <title>node chat via Socket</title>
        <script src="jquery-1.8.3.js"></script>
        <link href="chat.css" rel="stylesheet" type="text/css" />
    </head>
    <body>
        <h1 class="logo">欢迎进入通过Socket创建的聊天系统</h1>
        <div class="login" id="login_wrap">
        <!--登录窗口-->
            <h2>请先输入你的名字</h2>
            <input type="text" id="uname" class="txt"/>
            <!--先输入用户名-->
        </div>
        <div class="wrapper" id="wrapper">      <!--聊天窗口-->
            <div class="header">
                <h1 id = "page_title">通过Socket创建聊天室</h1>
            </div>
            <div class="content">
                <div class="chat_wrap">
                    <p class="chat_title">
                        欢迎<span id="J_uname"
 class="myname"></span>, <!-- 当前登录用户-->
                        当前时间是<span id="J_time"
class="time"></span>      <!-- 当前登录时间-->
                    </p>
                    <div class="chat_body">
                    <ul class="chat_msg" id="messages">
```

```
                              <!--对话消息列表-->
27                                      <li>欢迎进入聊天室，目前有
   <span class="userTotal"></span>人在聊天</li>
28                                  </ul>
29                                  <hr class="hr">
30                                  <div class="chat_act">
31                                          <label for="sendto">发送给
   </label>
32                                          <select name="sendto"
    id="sendto">              <!--发送对象用户列表 -->
33                                  <option value="all">所有人</option>
34                                          </select>
35                                      </div>
36                                      <div class="chat_input">
37                                              <!--聊天内容 -->
38                                          <textarea class=
   "chat_input_txt" id="chat_input"></textarea>
39                                      <div>
40                                  </div>
41                              </div>
42                          </div>
43                          <div class="chat_user">
44                              <p class="chat_title">用户列表</p>
45                              <div class="user_body">
46                                  <ul id="Userlist">  <!--用户列表 -->
47                                  </ul>
48                              </div>
49                          </div>
50                      </div>
51          <script src="chat.js"></script>         <!--引入chat.js文件 -->
52          </body>
53  </html>
```

第11行~第14行，当打开首页时，显示一个登录框，需要用户输入用户名，如图11.18所示。

第15行，聊天窗口的外层div，该div被默认隐藏，当用户输入了用户名后，该div被显示出来。

第22行~第23行，用户登录后，分别显示用户名和登录时间。

第26行~第28行，对话列表，当有新对话时包括用户间的对话和广播消息，都使用appendChild方法追加在ul元素内。

第32行~第34行，发送对象列表。当用户列表更新时，该列表也被更新，但该用户列表不包括登录用户自己。

第38行，消息发送窗口，按Enter键表示发送消息。

第46行~第47行，用户列表。当有新用户加入时，该列表被更新。

利用编辑器打开“www/chat.js”文件，代码如下：

```
01 if(!WebSocket){                              // 判断浏览器是否支持websocket
02       alert('Your browser does not support WebSocket, you cannot
   chat!');
```

```
}else{
var Socket;                         // 定义socket全局变量，程序运行时被赋值
var username;                       // 定义当前用户名全局变量
$(function(){
    var
    login_wrap = $("#login_wrap"),                  // 登录窗口
    uname_input = $("#uname"),                      // 登录用户名
    uname_welcome = $("#J_uname"),                  // 登录后显示用户名
    now_time = $("#J_time"),                        // 登录后显示当前时间
    userList = $("#Userlist"),                      // 用户列表
    userTotal = $(".userTotal"),                    // 用户总数
    sendto = $("#sendto"),                          // 发送对象用户列表
    message_list = $("#messages"),                  // 聊天消息列表
    chat_input = $('#chat_input'),                  // 发送消息文本
    chat_wrap = $("#wrapper");                      // 聊天窗口
    var now = function(){                           // 获取当前时间
        var time = new Date();
        return time.getHours() + ':' + time.getMinutes()
+ ':' + time.getSeconds();
    }
    var updateUserList = function(users){           // 更新用户列表
        var user_html = '',
            send_to_html = '<option value="all">所有人</option>';
        if(users.length > 0){
            users.forEach(function(user){
                user_html += '<li>' + user + '</li>';
                if(user != username){ // 加入聊天对象，不包括自己
                    send_to_html += '<option value="'+
 user +'">'+ user +'</option>'
                }
            });
            userList.html(user_html);               // 更新用户列表
            sendto.html(send_to_html);              // 更新发送对象
        }
        userTotal.html(users.length);               // 更新用户总数
    }
    var open = function(event){                 // socket连接成功回调函数
        login_wrap.remove();                        // 隐藏登录窗口
        chat_wrap.show();                           // 显示聊天窗口
        uname_welcome.text(username);               // 设置当前用户名
        now_time.text(new Date());                  // 设置当前时间
        var json_data = JSON.stringify({'message':username,
 'sender': 'ws:__join__'});
        Socket.send(json_data); // 发送消息socket消息，在服务器端注册用户
    };
    var message = function(event){    // 接收服务器端发回的消息回调函数
        var data = JSON.parse(event.data);    // 将消息转化为JSON格式
        var sender = data['sender'] || '';          // 消息的发送者
        var message = data['message'] || '';        // 消息内容
        if(sender == 'ws:__update_user__'){ // 发来的消息为更新用户列表动作
            updateUserList(message);    // 调用更新用户列表函数
```



第12章 离线Web应用大演练

本章的示例包含两大块技术内容：一块是Web Worker，用于解决JavaScript多线程的方案；另外一块是Cache Manifest，是一种HTML 5的缓存机制，可在没有因特网连接的情况下，照样使用Web应用，同时当恢复网络连接时，还能通过JavaScript自动对数据进行同步更新，这在HTML 5出现之前是一个完全无法想象的应用功能。另外，这对移动设备来说也是一项伟大的技术功能，在国内移动网络速度普遍缓慢的情况下，通过离线存储可以有效地缓解应用的访问速度问题。通过以下示例的学习，读者完全可以开发一款属于自己的离线应用，一起来学习吧。

本章知识点：

- Web Worker技术
- 离线存储技术
- 制作离线留言页面

 12.1 示例1 使用定时器

12.1.1 示例效果

本示例将用HTML 5的Web Worker重写第6章的动画计时器，读者可以通过对比两个示例更加直观地了解Web Worker的使用。首先，将页面文件和脚本部署在Web服务器上，如Apache、IIS或者Nginx等。

使用Chrome打开本示例页面地址，效果如图12.1所示。



图12.1 使用Chrome打开本示例页面地址

单击“开始”按钮开始计时，效果如图12.2所示。

```
33                interval = null;
34          } else if (type == 'reset') {                          // 复位操作
35                time_stop = 0;                                   // 清零时差
36                clearInterval(interval);
37                interval = null;
38                postMessage({ date: 0 });                        // 向页面传回初始数据
39          };
40    };
```

代码第01行~第19行分析可以参考第6章的“制作动画计时器”或者代码后方的注释。

代码第20行中的onmessage是self.onmessage或this.onmessage的缩写，self和this指的都是Worker线程的全局作用域，该方法用以监听主线程传入的消息。

在主线程中，操作按钮每次单击时将对应的操作类型字符串存入对象中，并传递给Worker线程，代码第21行~第22行中Worker线程接收到主线程返回的数据，取得数据中的操作类型，通过操作类型执行对应的业务操作逻辑。

代码第27行和第38行中的postMessage方法是self.postMessage或this.postMessage的缩写，该方法用于给主线程返回信息，可以接收字符串或JSON对象作为单个参数。



本例的演示只不过是HTML 5 Web Worker的冰山一角，Web Worker应用在目前还是新概念，更多的信息可以参考网站http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html。



12.2 示例2 排队处理订单

12.2.1 示例效果

本例展示一种常用的订单处理页面，不同之处在于订单会被发送至Web Worker线程进行处理，待处理完毕以后返回主线程。首先，将页面文件和脚本部署在Web服务器上，如Apache、IIS或者Nginx等。

使用Chrome打开本示例页面地址，效果如图12.3所示。

处理

□	订单号	商品名
□	1001	雨伞
□	1002	雨鞋
□	1003	雨衣
□	1004	雨帽
□	1005	雨裤

图12.3 使用Chrome打开本示例页面地址

选中首条订单号1001，单击“处理”按钮，订单数据被发送到Worker线程处理，处理完毕后返回“处理完毕”状态，效果如图12.4所示。

图12.4 处理订单1001号

选中剩余所有订单，单击“处理”按钮，订单将由上至下的顺序逐一被处理。

12.2.2 代码设计和分析

示例程序由主页面和Worker脚本两部分组成，主页面代码如下：

```
<!DOCTYPE HTML><html>
<head>
     <link rel="stylesheet" type="text/css" href="../css/grid.css" />
      <!-- 列表样式 -->
     <script src="../js/jquery-1.8.3.js"></script>
</head>
<body>
    <header><h2>排队处理订单</h2></header>
    <section>
       <div class="grid">
        <div class="window_search">
            <div class="window_tools">
                <div class="tDiv2">                 <!-- 操作按钮区 -->
                    <div class="fbutton"><div><span class="start">处理</span></div></div>
                    <div class="btnseparator"></div>
                </div>
            </div>
        </div>
        <div class="window_grid">                   <!-- 订单区 -->
            <table class="data tablesorter">
            <thead>
            <tr class="tablesorter-header">
                <th class="tablesorter-header">     <!-- 标题区 -->
                <div class="tablesorter-header-inner"><input type="checkbox" id="J_all" title="全选"/></div>
                </th>
```

```
25                  <th class="tablesorter-header"><div class="tablesorter-header-inner">订单号</div></th>
26                  <th class="tablesorter-header"><div class="tablesorter-header-inner">商品名</div></th>
27              </tr>
28              </thead>
29             <tbody>
30                  <tr class="zebra">              <!-- 商品列表信息区 -->
31                      <td><input type="checkbox" data-id="1001" /></td><td>1001</td><td>雨伞</td>
32                  </tr><!-- ...省略剩余商品结构，详见本书网络资源 -->
33              </tbody>
34              </table>                            <!-- 提示区 -->
35          </div><div class="window_page"><span class="btn J_tip"></span></div>
36      </div>
37      </section>
38 </body>
39 <script>
40      var check_all = $('#J_all'),                    // 全选按钮
41          check_items = $('input[data-id]'),          // 订单选项
42          tip = $('span.J_tip');                      // 提示框
43      check_all.on('click', function (e) {      // 监听全选按钮单击事件
44          check_items.attr('checked', this.checked);  // 订单选项与全选按钮同步
45      });
46      check_items.on('click', function (e) {    // 订单选项单击事件监听
47          check_all.attr('checked', !!$('input[data-id]:checked').length);
48      });
49      $('span.start').on('click', function (e) {    // “处理”按钮单击监听
50          var ids = [];
51          $('input[data-id]:checked').each(function (index, item) {  // 获取选中的订单号
52              ids.push($(item).attr('data-id'));
53          });
54          ids.length && work.postMessage({ ids: ids });  // 将订单号传给Web Worker处理
55      });
56      var work = new Worker('002.Worker.js');   // 创建一个Web Worker
57      work.onmessage = function (e) {           // 监听Web Worker消息事件
58          var data = e.data;
59          tip.html(data.msg);               // 获取返回的提示信息显示在提示区
60          $('input[data-id="' + data.id + '"]').closest('td').html('处理完毕');
61          check_all.attr('checked', false);     // 取消全选按钮选中状态
62      };
63 </script></html>
```

订单列表中，每条订单的复选框都带有“data-id”自定义属性，表示该条订单的订单号，见代码第31行。



12.3 示例3 在后台运行JavaScript

12.3.1 示例效果

本例将使用Web Worker在后台运行一段JavaScript，实现自动更新列表功能。本例后台数据存储将采用一个静态文本模拟，读者可以自行更改文本数据实现动态更新。首先，将页面文件和脚本部署在Web服务器上，如Apache、IIS或者Nginx等。

使用Chrome打开本示例页面地址，效果如图12.5所示。

姓名	性别	年龄
小王	女	31
小李	男	28
小周	男	29

图12.5 使用Chrome打开本示例页面地址

打开文件“data.json”，修改“小王”为“小张”，此时页面列表发生变化，效果如图12.6所示。

姓名	性别	年龄
小张	女	31
小李	男	28
小周	男	29

图12.6 修改“小王”为“小张”

再次打开文件“data.json”，新增一条JSON数据，新增数据格式如下：

```
{
    "name": "小刘",
    "age": 39,
    "sex": "女"
}
```

此时列表末尾出现新增信息，效果如图12.7所示。

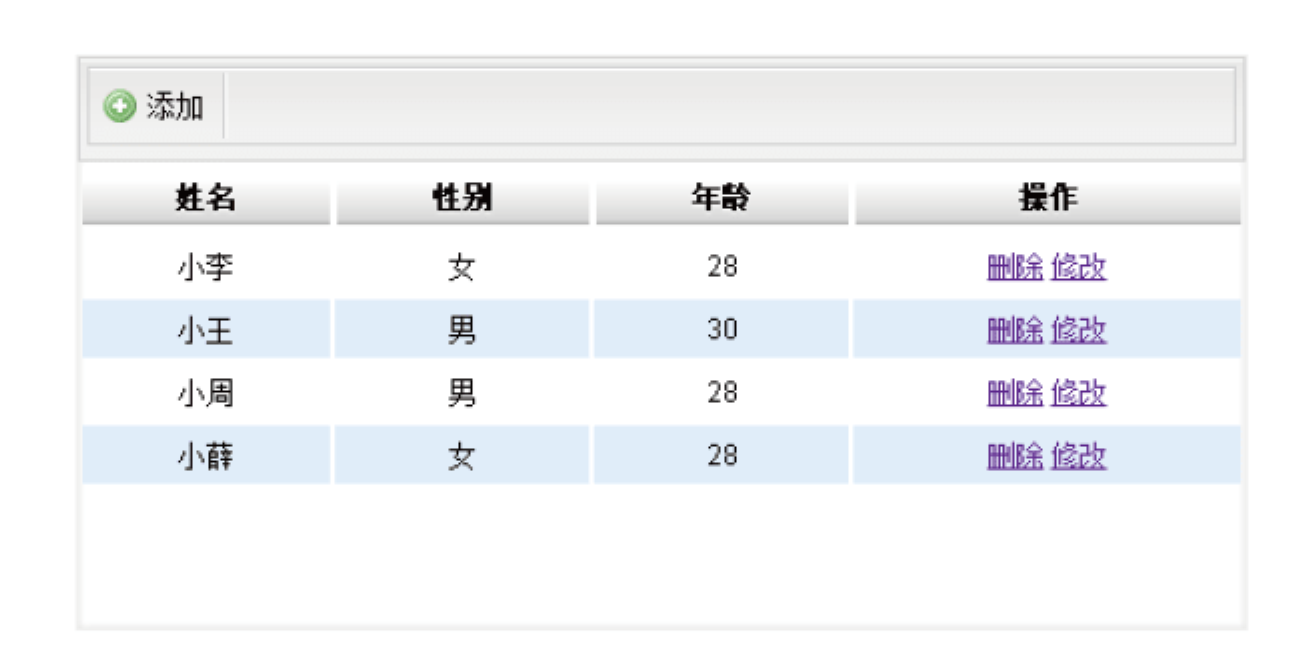
姓名	性别	年龄
小张	女	31
小李	男	28
小周	男	29
小刘	女	39

图12.7 新增一条信息

12.3.2 代码设计和分析

首先从展示页面开始分析，代码如下：

```
<!DOCTYPE HTML><html>
<head>
     <link rel="stylesheet" type="text/css" href="../css/grid.css" /> <!-- 列表样式 -->
     <script src="../js/jquery-1.8.3.js"></script>
</head>
<body>
    <header><h2>排队处理订单</h2></header>
    <section>
       <div class="grid">
        <div class="window_grid">
            <table class="data tablesorter">
                <thead>
                    <tr class="tablesorter-header">
                        <th class="tablesorter-header"> <!-- 标题结构 -->
                            <div class="tablesorter-header-inner">姓名</div>
                        </th>
                        <th class="tablesorter-header">
                            <div class="tablesorter-header-inner">性别</div>
                        </th>
                        <th class="tablesorter-header">
                            <div class="tablesorter-header-inner">年龄</div>
                        </th>
                    </tr>
                </thead><tbody></tbody>       <!-- 数据列表区 -->
            </table>
        </div>
    </div>
    </section>
    <script id="J_item" type="text/x-html5-tmpl">  <!-- 元素HTML模板 -->
    <tr class="zebra"><td>{name}</td><td>{sex}</td><td>{age}</td></tr>
```



添加

姓名	性别	年龄	操作
小李	女	28	删除 修改
小王	男	30	删除 修改
小周	男	28	删除 修改
小薛	女	28	删除 修改

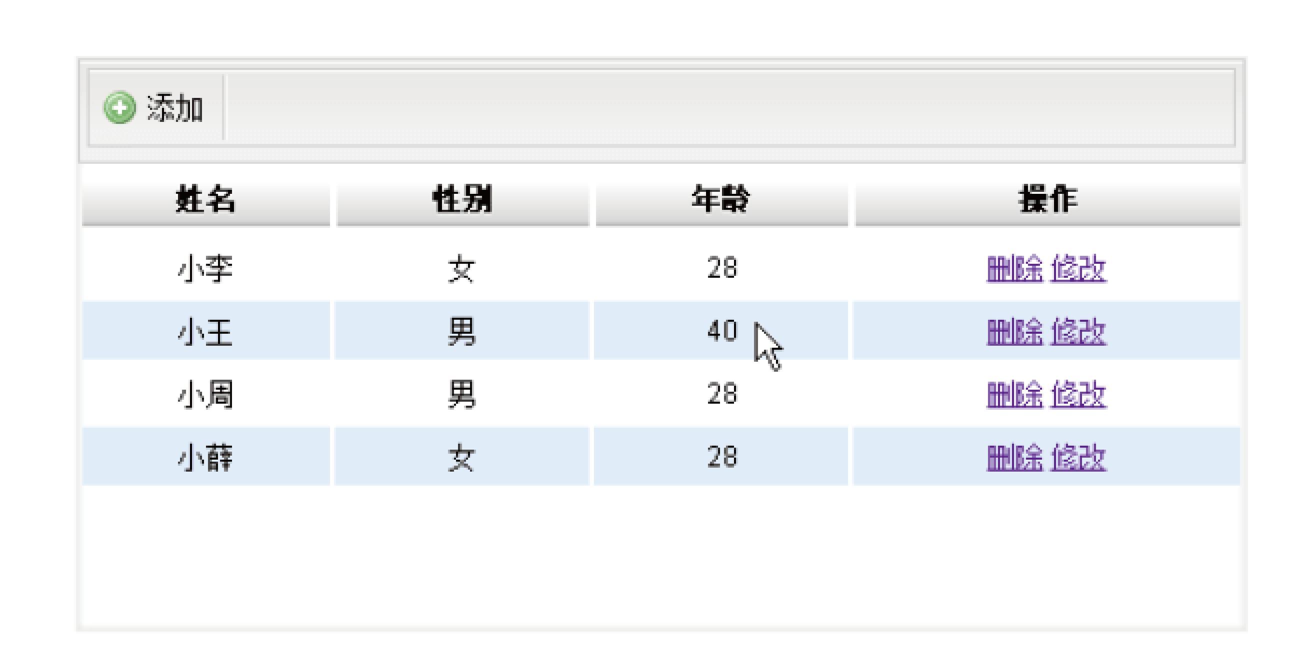
图12.9 添加多条用户信息

单击第2条数据“小王”操作区的修改按钮，出现带有“小王”录入信息的浮层，效果如图12.10所示。



图12.10 单击数据“小王”操作区修改按钮

修改“小王”的年龄为40，并单击“确认”按钮保存修改信息，此时列表对应内容同步更新，效果如图12.11所示。



添加

姓名	性别	年龄	操作
小李	女	28	删除 修改
小王	男	40	删除 修改
小周	男	28	删除 修改
小薛	女	28	删除 修改

图12.11 修改指定行信息

12.4.2 代码设计和分析

本示例功能的添加和删除功能已经在第10章的示例“使用本地数据库”进行了详细的介绍，本节将把介绍的重点放在新增的修改功能上。

列表元素模板区增加了一条修改按钮的HTML字符模板，元素的样式类名为modify，同时元素还带有一个自定义属性“data-id”用以表示该条记录的主键标识，模板代码如下：

```
<tr class="zebra">
    <td>{name}</td><td>{sex}</td><td>{age}</td>  // 姓名、性别、年龄信息
    <td>
        <a href="#" class="del" data-id="{id}">删除</a>          // 删除
        <a href="#" class="modify" data-id="{id}">修改</a>       // 修改按钮
    </td>
</tr>
```

独立出方法open_dialog，为用户弹出信息录入框，代码如下：

```
function open_dialog(title, callback) {
    var wraper = $($('#J_form').html());                 // 新建添加弹层
    $(document.body).append(wraper);                     // 注意页面结构
    wraper.dialog({
        position: 'center', title: title, modal: true,
        buttons: [{ text: "确认",
            click: function () {
                var name = wraper.find('input.name').val(),      // 姓名值
                    sex = wraper.find('select.sex').val(),       // 性别值
                    age = wraper.find('input.age').val();        // 年龄值
                if (name.length && sex.length && age.length) {  // 输入验证
                    callback ({ name: name, sex: sex, age: age });
                    // 执行回调函数，传入录入信息
                    $(this).dialog("close");                     // 关闭弹出
                } else { alert('请正确填写所有填写内容');  };
            }
        }, { text: "取消",
            click: function () { $(this).dialog("close"); }  // 关闭按钮事件
        }]
    });
    wraper.dialog('open');                                       // 打开弹出框
    return wraper;
};
```

open_dialog函数接收两个参数，说明如下。

- title：弹出框的标题，字符型。
- callback：单击“确认”按钮后的回调函数，接收一个数据对象。

新增的另外一个函数modify_item，用以完成弹出修改框，将确认的内容同步更新至本地离线数据库，代码如下：

```
function modify_item(target) {
    var id = target.attr('data-id');                     // 该条记录的主键标识
    function update_db(data) {                           // 更新本例离线数据库
        storageDriver.transaction(function (t) {
            t.executeSql("update " + DB_NAME + " set name=?, sex=?,
    age=? where id=?",
```

```
                    [data.name, data.sex, data.age, id],
                    // 传入修改数据
                    function (transaction, resultSet) {
                        target.closest('tr').replaceWith(substitute(item_tpl,
        data));// 更新对应行信息
        }); }); };
        storageDriver.transaction(function (t) {
            t.executeSql("SELECT * FROM " + DB_NAME + ' where id=' + id, [],
            // 读指定行信息
                function (t, results) {
                    var wraper = open_dialog('修改', function (data) {
                    // 弹出修改框
                        data.id = id;                   // 记录信息主键值
                        update_db(data);                // 存入本地离线数据库
                    }), result;
                    for (var i = 0, l = results.rows.length; i < l; i++) {
                        result = results.rows.item(i);
                        wraper.find('input.name').val(result.name);
                        // 设置姓名
                        wraper.find('select.sex').val(result.sex);
                        // 设置性别
                        wraper.find('input.age').val(result.age);
                        // 设置年龄
                    };
                });
        });
    };
```

modify_item方法一共会完成两条SQL语句。首先，从本地离线数据库中读取指定行信息并填充至弹出框的对应输入框内。单击修改框中的“确认”按钮，会触发该方法的内部函数update_db，该函数接收一个存放用户信息的数据对象参数，修改数据库执行Update语句，SQL语法如下：

```
    UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值
```

最后，不要忘记在列表的监听代理上添加对修改按钮单击事件的捕捉，代码如下：

```
    tbody.on('click', function (e) {                    // 监听列表单击事件
        var target = $(e.target);
        if (target.hasClass('del')) {                   // 目标元素为删除按钮
            e.preventDefault();                         // 阻止a元素默认事件
            del_item(target);                           // 删除元素和数据库内容
        } else if (target.hasClass('modify')) {         // 目标元素为修改按钮
            e.preventDefault();
            modify_item(target);                        // 弹出修改浮层
        };
    });
```

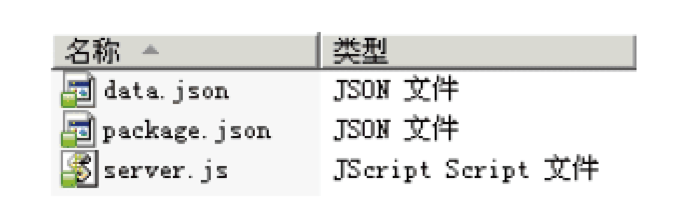
通过目标元素的样式类名判断元素的操作类型，当目标元素含有modify的样式类名时，即表示为操作按钮触发，此时调用modify_item方法弹出修改浮层，执行修改相关操作。

12.5 示例5 检测网络的当前状态

12.5.1 示例效果

传统的Web应用在遇到没有网络的状况下基本无法使用，用户的客户端与远端的服务器将失去网络这个纽带。HTML 5带来了一个似乎完美的解决方案。本示例将在第9章的示例“对照片进行排序”的基础上增加离线存储和联网同步的功能，实现不丢失任何信息的一次排序操作。

首先，进入本示例的后端Server文件夹“file-server”，里面存放了三个文件，“data.json”用于存放商品列表数组的字符序列化数据，“package.json”存放了模块的描述信息，server.js”存放了主要的存储和读取逻辑脚本，如图12.12所示。



名称	类型
data.json	JSON 文件
package.json	JSON 文件
server.js	JScript Script 文件

图12.12 后端Server文件夹“file-server”

在Windows下打开命令行进入刚才的目录，执行代码如下：

```
node server.js
```

如果启动成功，命令行提示如下：

```
listening on http://localhost:8080
```

使用Chrome浏览器打开文件，页面加载完毕后向后端服务器发送一个Ajax请求（地址为http://localhost:8080），获取商品列表信息并渲染至网页上，效果如图12.13所示。



图12.13 使用Chrome打开文件

此时，计算机处于联网状态，拖动图片“护目镜”至第一位，拖动完毕时页面发送商品最新排序数据至后端服务器，发送数据如图12.14所示。

```
Headers  Preview  Response  Timing
Request URL: http://localhost:8080/
Request Method: POST
Status Code: 200 OK
▼Request Headers    view source
  Accept: */*
  Accept-Encoding: gzip,deflate,sdch
  Accept-Language: zh-CN,zh;q=0.8
  Connection: keep-alive
  Content-Length: 233
  Content-Type: application/x-www-form-urlencoded
  Host: localhost:8080
  Origin: null
  User-Agent: Mozilla/5.0 (Windows NT 5.2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.94 Safari/537.36
▼Form Data    view source    view URL encoded
  type: set
  data: ["护目镜","男套装","黑色护腕","护腿","篮球鞋","太阳镜"]
▼Response Headers    view source
  Access-Control-Allow-Methods: GET, POST
  Access-Control-Allow-Origin: *
  Connection: keep-alive
  Date: Sat, 25 May 2013 10:41:24 GMT
  Transfer-Encoding: chunked
```

图12.14 存储拖曳后商品列表信息请求数据

断开本地网络使计算机处于离线状态，拖曳图片“太阳镜”至第一位，程序会将商品列表信息存储于LocalStorage中，当计算机再次联网时，自动同步至远端服务器。LocalSorage数据如图12.15所示。

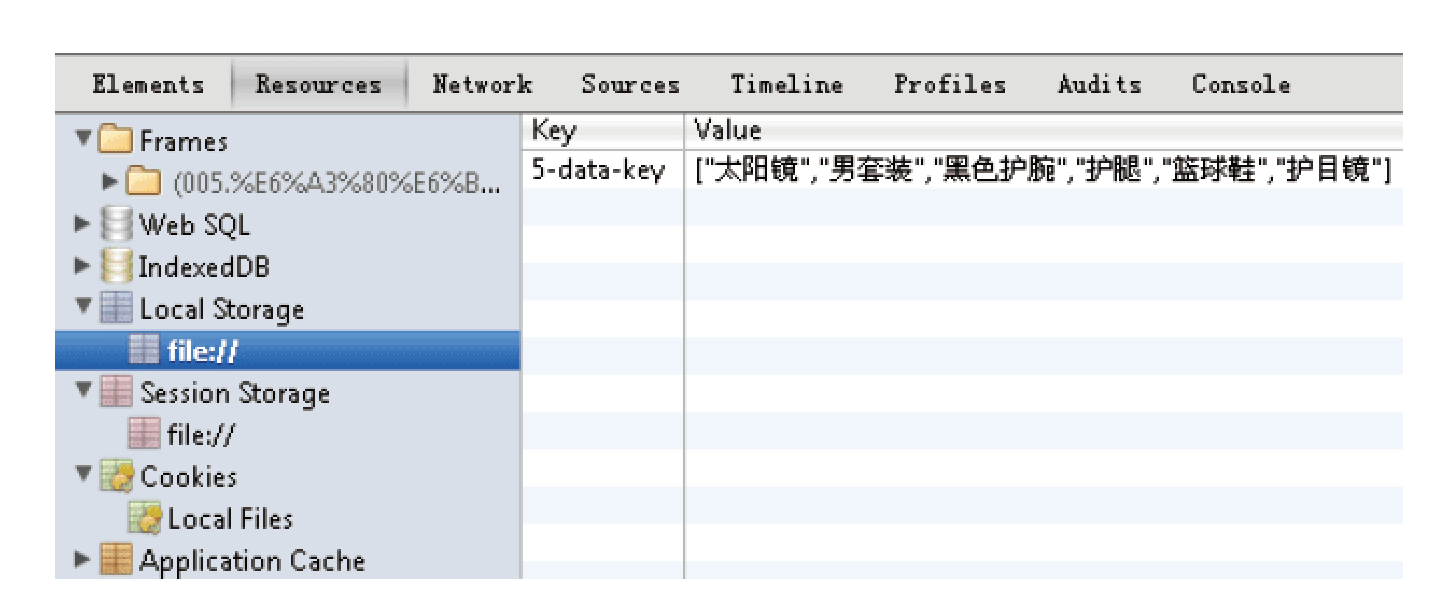


图12.15 计算机离线状态LocalSorage存储信息

12.5.2 代码设计和分析

拖曳功能的实现已经在第9章的“对照片进行排序”示例进行了详细的介绍，本节将对前后端存储和网络检测部分进行介绍分析。

首先分析客户端网页代码，其中一个非常重要的方法store_item_position，用于存储每次拖曳后商品列表的数据信息，代码如下：

```
function store_item_position(data) {
    if (navigator.onLine) {                                  // 是否联网在线
        var xmlhttp = new XMLHttpRequest();      // 新建XMLHttpRequest实例
        xmlhttp.onreadystatechange = function () {   // 监听请求状态变化
            if (xmlhttp.readyState == 4) {      // 判断请求是否加载完毕
```

```
06                  if (xmlhttp.status == 200) {         // 判断请求是否成功
07                      var data = JSON.parse(xmlhttp.responseText);
                        // 将获取对象解析为JSON
08                      if (data.code == 200) {
09                          localStorage.removeItem(STORAGE_KEY);
                            // 移除本地离线存储
10                      };
11                  };
12              };
13          };                                           // 异步请求远端数据
14          xmlhttp.open("post", 'http://localhost:8080', true);
15          xmlhttp.setRequestHeader("Content-Type", "application/x-www-
   form-urlencoded")
16          xmlhttp.send('type=set&data=' + encodeURIComponent(JSON.
   stringify(data)));
17      } else {                                         // 非联网在线
18          localStorage.setItem(STORAGE_KEY, JSON.stringify(data));
            // 本地离线存储列表数据
19      };
20  };
```

该方法第02行，使用navigator对象的onLine属性判断当前的网络状态。当计算机处于联网状态，将商品列表数据data（见代码第16行）发送至后端服务器进行存储。如果存储成功，调用LocalStorage的removeItem方法清空用于本地存储的缓存键指，见该方法第09行。

当navigator对象的onLine属性为false，此时计算机处理离线状态，数据无法保存至远端的存储服务器，所以采取将商品列表数据存储至本地浏览器的LocalStorage中，见代码第18行。

函数build_list用于构建商品列表的HTML结构，同时绑定拖曳相关事件，代码如下：

```
function build_list(data) {
    var htmls = [];                              // 存储列表结构数组
    data.forEach(function (item) {               // 循环构建商品列表HTML结构
        htmls.push('<div draggable="true"><img src="../images/shop/'
   + item + '.png" draggable="false"/>' + item + '</div>');
    });
    document.querySelector('section').innerHTML = htmls.join('');
   // 填充商品列表容器
    bind_event(slice.call(document.getElementsByTagName('div'), 0));
   // 绑定拖曳相关事件
};
```

除了上面介绍的两个函数，每当用户进入页面时会执行渲染商品列表相关的逻辑，代码如下：

```
var STORAGE_KEY = '5-data-key',                          // 本地缓存键值
    storage_data = localStorage.getItem(STORAGE_KEY); // 本地缓存数据
if (storage_data) {                                      // 存在离线内容
    storage_data = JSON.parse(storage_data);             // 解析为JSON对象
    build_list(storage_data);                            // 生成列表信息
    store_item_position(storage_data);                   // 存储后端离线信息
```

```
08      res.setHeader('Access-Control-Allow-Methods', 'GET, POST');
09      req.setEncoding('utf8');            // 设置接收数据编码格式为 UTF-8
10      req.addListener('data', function (chunk) {
        // 接收数据块并将其赋值给 postData
11          postData += chunk;
12      }).addListener('end', function () {
13          postData = querystring.parse(postData);
            // 解析为URL参数对象
14          var result;
15          if (postData.type == 'set') {                  // 存储商品列表信息
16              fs.writeFile(__dirname + '\\data.json',
    JSON.stringify(postData.data || []), function (err) {
17                  if (err) { result = { "code": 500 };
18                  } else { result = { "code": 200 }; };
19                  res.end(JSON.stringify(result));     // 返回存储状态
20              });
21          } else {                                       // 获取商品列表信息
22              fs.readFile(__dirname + '\\data.json', 'utf8',
     function (err, data) {
23                  if (err) { esult = { "code": 500 };
24                  } else { result = { "code": 200, "data":
    JSON.parse(data) }; };
25                  res.end(JSON.stringify(result));
                    // 返回商品数组序列化字符串
26              });
27          };
28      });
29  }).listen(8080, function () {            // 设置Web服务器监听端口，并启动服务
30      console.log('listening on http://localhost:8080');
            // 控制台显示Web服务器启动成功
31  });
```

服务器端主要包含读取数据和存储数据两块功能。当从URL参数中获取参数type类型为set时，表示执行存储操作，见代码第15行~第20行，调用fs模块的writeFie方法，该方法语法如下：

```
fs.writeFile(filename, data, [options], callback)
```

- filename：文件完整路径。
- data：存储数据，string或者buffer。
- options：可选参数对象，可忽略。
- callback：回调函数。

当参数type为其他时，表示获取商品列表数据，见代码第22行~第26行，调用fs模块的readFile方法，该方法语法如下：

```
fs.readFile(filename, [options], callback)
```

- filename：文件完整路径。
- options：可选参数对象，可忽略。
- callback：回调函数。

商品列表存储文件（即data.json）数据格式如下：

```
"[\"护目镜\",\"男套装\",\"黑色护腕\",\"护腿\",\"篮球鞋\",\"太阳镜\"]"
```



12.6 示例6 开发离线留言网页

12.6.1 示例效果

本例将结合之前所用的离线应用知识，实现一个可在离线环境使用的留言网页，进一步展现Web SQL在日常开发中的使用。

使用Chrome浏览器打开网页文件，运行效果如图12.16所示。

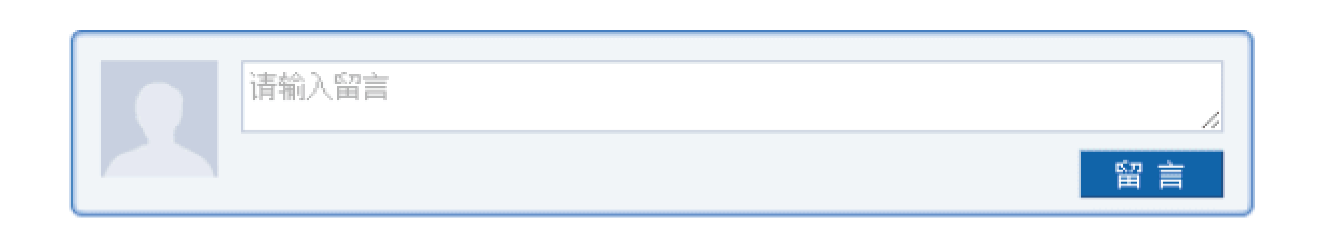


图12.16 使用Chrome打开网页文件

在没有输入任何内容时，单击“留言”按钮，输入框下方会出现红色的提示信息“请填写留言内容”，效果如图12.17所示。

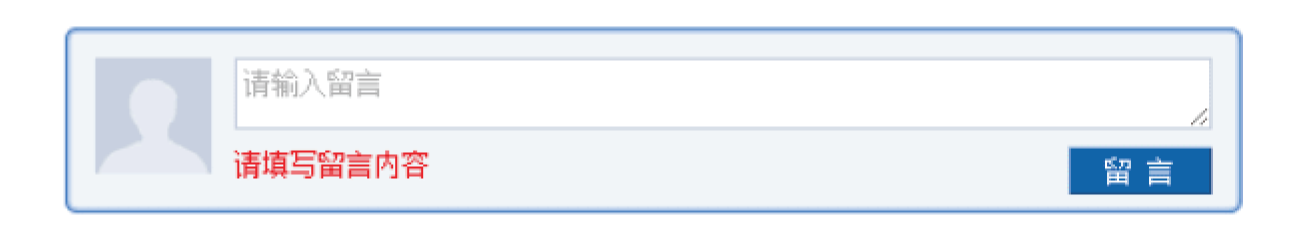


图12.17 不输入内容直接单击“留言”按钮

在输入框中输入内容“开发一个离线留言网页”，单击“留言”按钮，提交成功后下方会出现刚才输入的留言信息，并附上随机生成的头像和用户名，效果如图12.18所示。



图12.18 输入信息并单击“留言”按钮

重复多次刚才的留言操作，效果如图12.19所示。

show_error_tip函数用于实现错误提示功能，并在提示出现后的1500ms自动隐藏提示信息。留言按钮的单击事件监听由代码第72行~第82行完成，单击按钮后，首先会判断在输入框中是否输入了留言信息，输入内容会被trim方法去除头尾的空格符，确保输入内容真实有效。

每次刷新页面，脚本都会自动从Web SQL中读取历史留言信息并进行渲染，同时对于第1次进入页面的用户，会在该用户的浏览器中自动创建一张用于存放留言信息的数据表，见代码第84行~第97行。



本例中使用的Web SQL还只是HTML 5中的冰山一角，浏览器客户端数据库还有更多的功能等待读者去发现，详情可参考网站http://www.w3.org/TR/webdatabase/。



12.7 示例7 添加Geolocation跟踪

12.7.1 示例效果

时常在手机的APP应用中能够看到用户发送留言都会跟着当前的地理位置信息，现在由于HTML 5的到来，在网页上实现这一功能也是轻而易举的事情。本例将结合上一示例“开发一个离线留言网页”，增加动态获取用户地理信息的功能。首先，将页面文件和脚本部署在Web服务器上，如Apache、IIS或者Nginx等。

使用Chrome浏览器打开本示例页面地址，效果如图12.20所示。

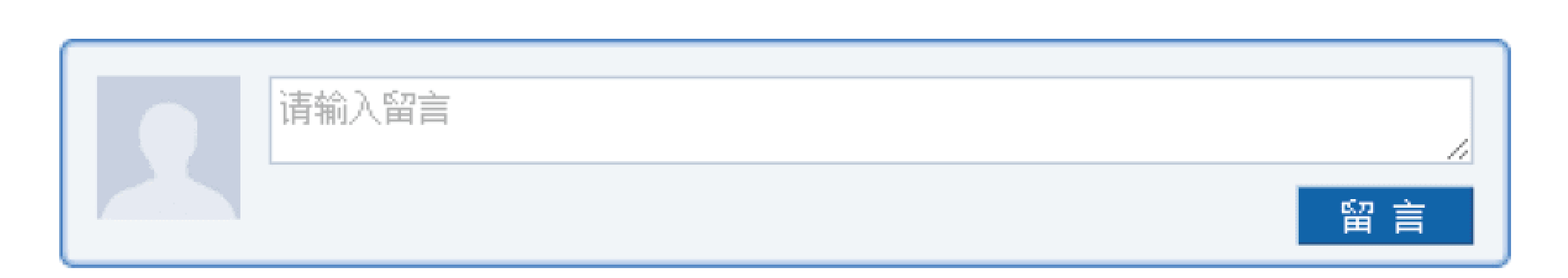


图12.20 使用Chrome浏览器打开本示例页面地址

然后，进入本示例后端服务器文件夹“geolocation-server”，一共存放了两个文件，“package.json”存放了模块的描述信息，“server.js”存放了主要的存储和读取逻辑脚本，如图12.21所示。



图12.21 后端服务器“geolocation-server”文件夹

在Windows下打开命令行进入刚才的目录，执行代码如下：

```
node server.js
```

如果启动成功，命令行提示如下：

```
listening on http://localhost:8080
```

在输入框中输入内容“添加Geolocation跟踪”，单击“留言”按钮，提交成功后下方会出现刚才输入的留言信息，并附上随机生成的头像和用户名，同时还带有当前的地理位置信息，效果如图12.22所示。



图12.22 输入信息并单击“留言”按钮

重复多次刚才的留言操作，效果如图12.23所示。



图12.23 多次重复留言操作

12.7.2 代码设计和分析

HTML 5本身不会提供完整的地理位置信息，但是会返回关键的经纬度信息，然后开发人员可以通过谷歌的公共地理服务将经纬度转换成对应的地理位置信息。

首先，要对数据表进行修改，增加地理位置信息geolocation字段，代码如下：

```
t.executeSql("CREATE TABLE IF NOT EXISTS " + DB_NAME +   // 创建数据表
            "(id INTEGER PRIMARY KEY AUTOINCREMENT, "+   // 自增字段
            "name TEXT NOT NULL, " +                     // 姓名字段
            "date TEXT NOT NULL, " +                     // 时间字段
            "content TEXT NOT NULL, " +                  // 内容字段
            "geolocation TEXT NOT NULL, " +              // 地理信息字段
            "img INTEGER DEFAULT 1)");                   // 照片字段
```

监听留言按钮的单击事件，增加获取用户经纬度和转换经纬度为地理位置信息的功能，代码如下：

```
submit_btn.addEventListener('click', function (e) {
```

```
02      e.preventDefault();
03      var content = textarea_el.value.trim();
04      if (content.length) {
05          navigator.geolocation.getCurrentPosition(function (position) {
                  // 获取经纬度信息回调
06              var xmlhttp = new XMLHttpRequest(); // 新建XMLHttpRequest实例
07              xmlhttp.onreadystatechange = function () {
                      // 监听请求状态变化
08                  if (xmlhttp.readyState == 4) {  // 判断请求是否加载完毕
09                      if (xmlhttp.status == 200) {     // 判断请求是否成功
10                          var data = JSON.parse(xmlhttp.responseText);
                            // 将获取对象解析为JSON
11                          store_data({
12                              img: (new Date().getTime()) % 5,
                                 // 随机头像
13                              name: '陌生人' + (new Date().getTime()) % 5,
                                 // 随机昵称
14                              date: new Date().toLocaleString(),
15                              content: content,
16                                // 返回市一级别的地理位置信息
17                              geolocation: data.results[data.results.
   length - 3].formatted_address
18                }); }; };};
19              xmlhttp.open("post", 'http://localhost:8080', true);
                // 异步请求远端数据
20              xmlhttp.setRequestHeader("Content-Type", "application/
   x-www-form-urlencoded");
21              // 获取经纬度，并传递给Ajax请求
22              xmlhttp.send('lat=' + position.coords.latitude + '&lon=' +
   position.coords.longitude);
23          });
24      } else { show_error_tip('请填写留言内容'); };
25  }, false)
```

经纬度和地址的转换接口由于跨域的关系无法用Ajax直接进行调用，这里采用通过后端程序进行包装提供调用接口。navigator.geolocation的getCurrentPosition方法接收一个匿名函数，当浏览器获得经纬度信息时产生回调，并通过函数接收数据position，position的coords属性可以获取经纬度信息，见代码第05行和第22行。

下面分析后端Node.js服务，该服务主要完成调用远程谷歌地理服务的工作，代码如下：

```
var querystring = require('querystring'),     // 浏览器参数数据获取模块
    http = require("http");                   // 引用http模块，用于Web服务器
http.createServer(function (req, res) {       // 创建新服务器
    var postData = '';
    res.setHeader('Access-Control-Allow-Origin', '*');
    // 所有域名跨域访问均可以被通过
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST');
    // 服务器支持' GET, POST'方法
    req.setEncoding('utf8');              // 设置接收数据编码格式为UTF-8
    req.addListener('data', function (chunk) {
```

```
09          postData += chunk;
10      }).addListener('end', function () {
11          postData = querystring.parse(postData);
12          http.get("http://maps.google.com/maps/api/geocode/json?latlng=" +
            // Google地理位置服务
13                    postData.lat + "," + postData.lon +
                      // 接收的经纬度信息
14                    "&language=zh-CN&sensor=true", function (_res) {
                      // 请求回调函数
15                        var result = '';
16                        _res.setEncoding('utf8');       // 设置请求编码
17                        _res.on('data', function (chunk) { result
    += chunk; });       // 获取请求数据
18                        _res.on('end', function () { res.end(result); });
                          // 请求结束返回结果
19                    });
20      });
21  }).listen(8080, function () {                          // 启动服务
22      console.log('listening on http://localhost:8080');
        // Web服务器启动成功
23  });
```

Node.js服务获取客户端请求传送的经纬度数据，并将该数据传递给谷歌地理位置服务接口，接口分析数据后将对应的地理位置信息返回，服务接收到完成的数据后将该数据发送给客户端，见代码第12行~第19行。



提示 谷歌地理位置服务请参考官网文档https://developers.google.com/maps/documentation/geocoding/。



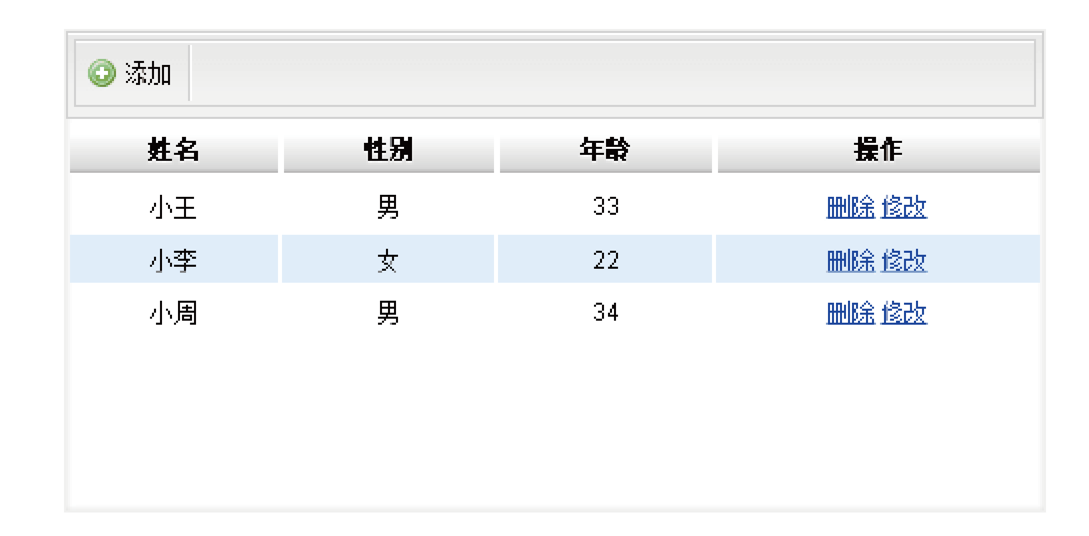
12.8 示例8 设计离线事件处理程序

12.8.1 示例效果

现今的Web世界正在向传统桌面应用挑战，页面变得越来越复杂，功能也变得越来越多，SPA（Single Page Application）独立页面应用开发也变得稀疏平常，非常具有代表性的应用如谷歌的Gmail、腾讯的WebQQ，众多的功能被集中在一个页面完成。面对如此庞大的功能集，传统页面的加载性能面临着巨大的挑战。HTML 5带来了Application Cache API，提供了一系列新特性支持离线引用缓存。下面通过在示例“开发简单的离线应用”的基础上添加离线事件处理功能，介绍Application Cache API相关的技术和知识点。

首先，将本示例页面文件和脚本部署在Web服务器上，如Apache、IIS或者Nginx等。使

用Chrome打开本示例页面地址，并添加若干条信息，效果如图12.24所示。



添加

姓名	性别	年龄	操作
小王	男	33	删除 修改
小李	女	22	删除 修改
小周	男	34	删除 修改

图12.24　使用Chrome打开本示例页面地址

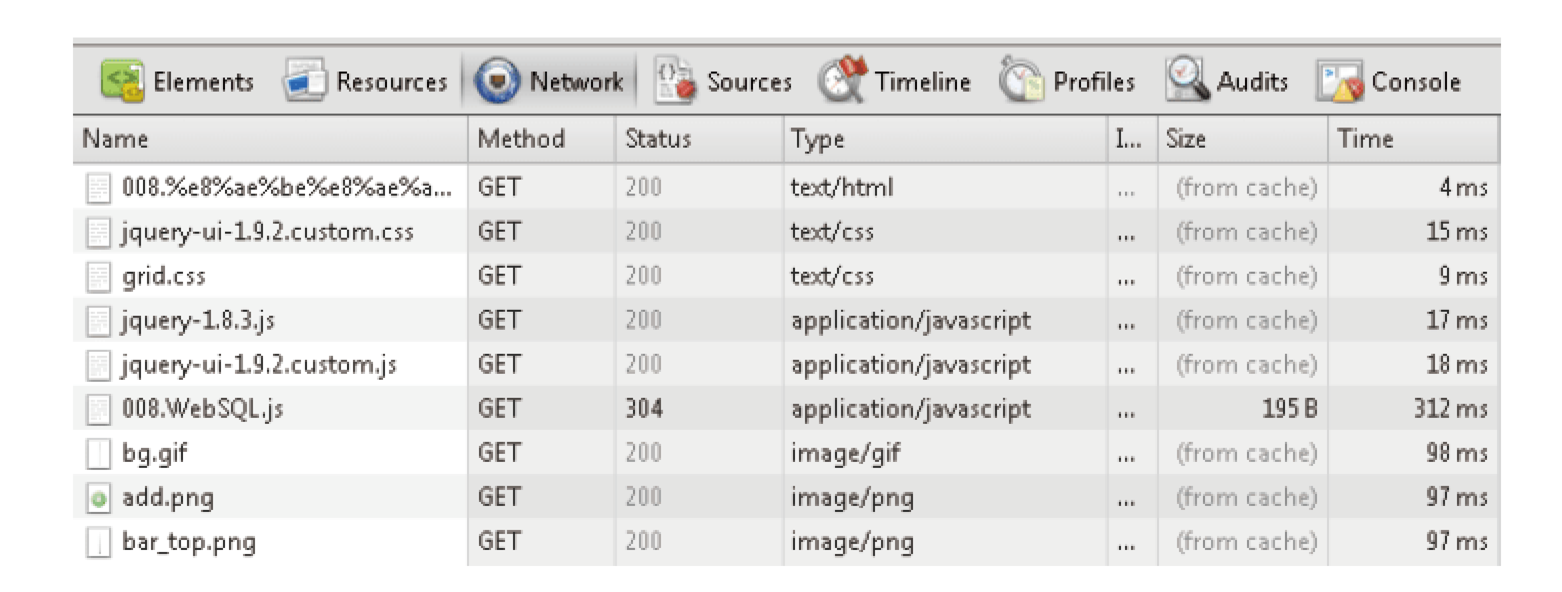
首次打开页面，所有请求均获得服务器响应的HTTP状态200信息，如图12.25所示。

Elements　Resources　Network　Sources　Timeline　Profiles　Audits　Console

Name	Method	Status	Type	I...	Size	Time
008.%e8%ae%be%e8%ae%a...	GET	200	text/html	...	3.9 KB	311 ms
jquery-ui-1.9.2.custom.css	GET	200	text/css	...	32.4 KB	63 ms
grid.css	GET	200	text/css	...	3.5 KB	67 ms
jquery-1.8.3.js	GET	200	application/javascript	...	262 KB	105 ms
jquery-ui-1.9.2.custom.js	GET	200	application/javascript	...	454 KB	123 ms
008.WebSQL.js	GET	200	application/javascript	...	6.4 KB	171 ms
bar_top.png	GET	200	image/png	...	484 B	131 ms
bg.gif	GET	200	image/gif	...	1.1 KB	130 ms
add.png	GET	200	image/png	...	1.1 KB	126 ms

图12.25　页面

刷新当前页面，部分请求的Size区域标识为from cache表示该条请求数据从浏览器内部缓存中获取，可以非常明显地看到请求耗时（Time）骤减，如图12.26所示。



Elements　Resources　Network　Sources　Timeline　Profiles　Audits　Console

Name	Method	Status	Type	I...	Size	Time
008.%e8%ae%be%e8%ae%a...	GET	200	text/html	...	(from cache)	4 ms
jquery-ui-1.9.2.custom.css	GET	200	text/css	...	(from cache)	15 ms
grid.css	GET	200	text/css	...	(from cache)	9 ms
jquery-1.8.3.js	GET	200	application/javascript	...	(from cache)	17 ms
jquery-ui-1.9.2.custom.js	GET	200	application/javascript	...	(from cache)	18 ms
008.WebSQL.js	GET	304	application/javascript	...	195 B	312 ms
bg.gif	GET	200	image/gif	...	(from cache)	98 ms
add.png	GET	200	image/png	...	(from cache)	97 ms
bar_top.png	GET	200	image/png	...	(from cache)	97 ms

图12.26　刷新当前页面

更新appcache.manifest文件，使用“#”符号注释“../js/jquery-ui-1.9.2.custom.js”，保存后关闭，修改后的代码如下：

```
CACHE MANIFEST

CACHE:

#../js/jquery-ui-1.9.2.custom.js
../js/jquery-1.8.3.js
```

```
15              if (window.applicationCache.status ==
   window.applicationCache.UPDATEREADY) {
16                  window.applicationCache.swapCache();    // 替换旧缓存内容
17                  var wraper = $('<p>是否更新缓存文件？</p>');
18                  $(document.body).append(wraper);        // 添加弹出节点
19                  wraper.dialog({                         // 弹出提示浮层
20                      position: 'center', title: '提示', modal: true,
21                      buttons: [{ text: "确认",
22                          click: function () { window.location.reload(); }
                            // 刷新当前页面
23                      }, { text: "取消",
24                          click: function () { $(this).dialog("close"); }
                            // 关闭按钮事件
25                      }]
26                  });
27                  wraper.dialog('open');                // 弹出复测
28              };
29          }, false);
30      }, false);
31  </script></html>
```

首先注意到，在HTML的起始节点上增加了manifest属性，可以通过相对或绝对路径设置，但必须保证manifest文件与当前页面处于同一域名下。manifest指向的文件可以是任意的后缀，但是MIME Type必须是text/cache-manifest。如果使用的是Apache服务器，可以在Apache程序目录的conf文件夹内找到mime.types文件，并添加一行信息，用于对manifest后缀文件的支持，代码如下：

```
text/cache-manifest                              manifest
```

本示例manifest文件appcache.manifest的代码如下：

```
CACHE MANIFEST
CACHE:
../js/jquery-ui-1.9.2.custom.js
../js/jquery-1.8.3.js
../css/grid.css
../css/jquery-ui-1.9.2.custom.css
../css/images/grid/bg.gif
../css/images/grid/add.png
../css/images/grid/bar_top.png
NETWORK:
*
```

文件以CACHE MANIFEST开头，CACHE下面列举的文件在第一次访问以后会被浏览器缓存，NETWORK下列举的文件表示除CACHE内的均通过网络下载，均支持通配符选择。



Chrome下可以通过访问chrome://appcache-internals/来查看或者清除离线缓存数据。

下面分析页面中的脚本逻辑，见代码第12行~第30行。

代码第13行监听离线缓存对象applicationCache的updateready事件，当manifest文件产生变

HTML 5手机遥控PPT



本章将展示一个通过在手机界面上进行左右滑动来远端操控多个PPT的翻动效果。整个实例运用了多种新技术框架，如HTML 5的WebSocket、CSS 3的动画、Node.js的Web应用框架Express、Node.js的通信类库Socket.IO等。本章将结合新技术和使用场景，向读者介绍不同技术在现实场景中的运作机制。

本章知识点：

- 使用移动设备访问控制器页面
- index路由的逻辑规则
- handle路由的逻辑规则
- Consolidate.js库



13.1 控制器页面预览

首先，确保本机已经安装Node.js（可以参考第4章的环境搭建），打开本例对应的代码文件夹，如图13.1所示，其中各文件夹的意义如下：

- server.js存放主要功能的逻辑脚本。
- package.json存放模块的描述信息。
- node_modules文件夹用于存放程序依赖模块。
- public文件夹用于存放页面用到的图片、样式和脚本。
- routes文件夹用于存放请求路由的脚本逻辑文件。
- views文件夹用于存放静态页面的模板文件。



图13.1 实例代码文件

在Windows系统下打开命令行进入刚才的文件夹，执行命令如下：

```
node server.js
```

如果启动成功，命令行提示如下：

```
   info  - socket.io started
Express server listening on port 3000
```

使用Chrome浏览器访问地址“http://localhost:3000/#/bored”，页面信息内容为受“菲特”台风影响的余姚市的相关介绍，PPT使用Impress.js类库（该类库使用CSS 3D Transforms的旋转、扭曲、缩放等特性制作，它是供开发者使用的表现层演示工具），效果如图13.2所示。



图13.2 PPT显示端页面效果

使用Chrome浏览器访问地址“http://localhost:3000/handle”，该页面为PPT的控制器，如果是非触屏设备的浏览器访问，可以通过单击页面，同时向左向后滑动对显示端的PPT进行前后翻页切换，控制器页面效果如图13.3所示。



图13.3 控制器页面

单击控制器页面中心区域，此时出现一个红色方块，向右拖动后松开鼠标，效果如图13.4所示。

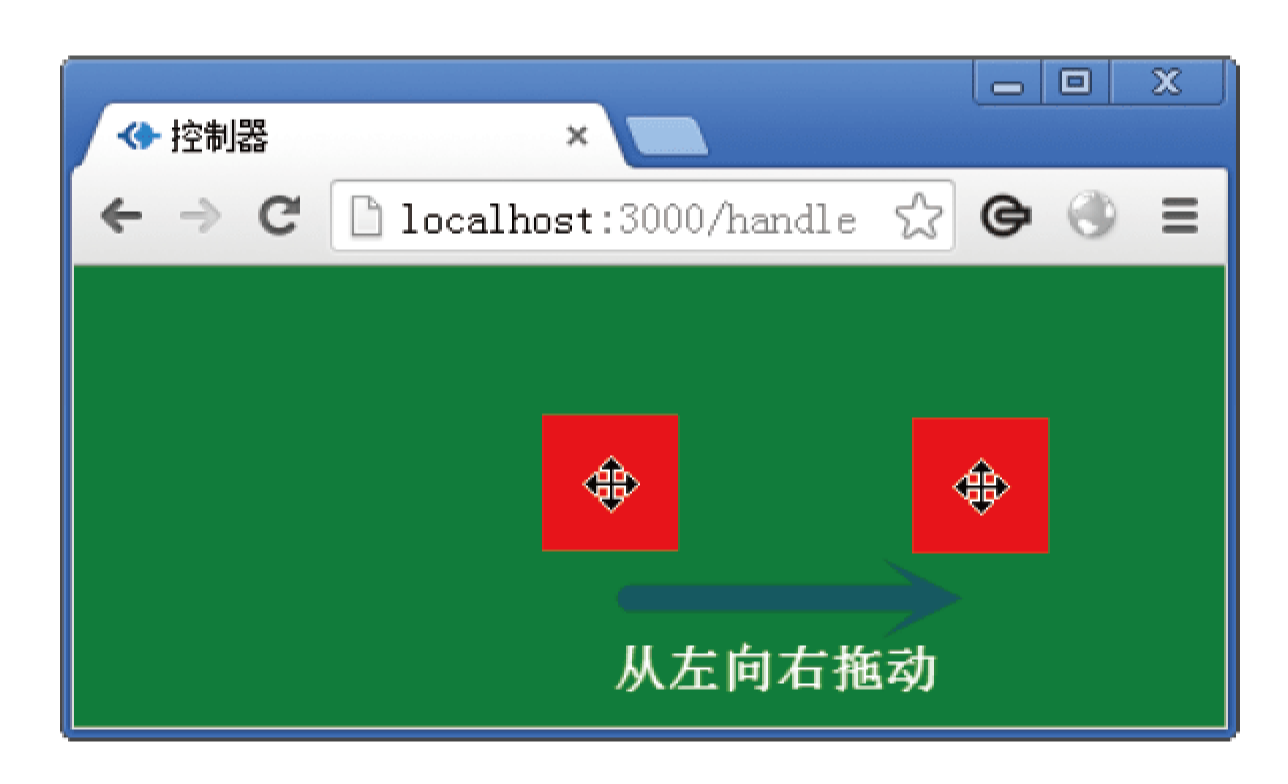


图13.4　单击控制器页面并向右拖动鼠标

查看主PPT页面，幻灯片由第1张切换至第2张，效果如图13.5所示。

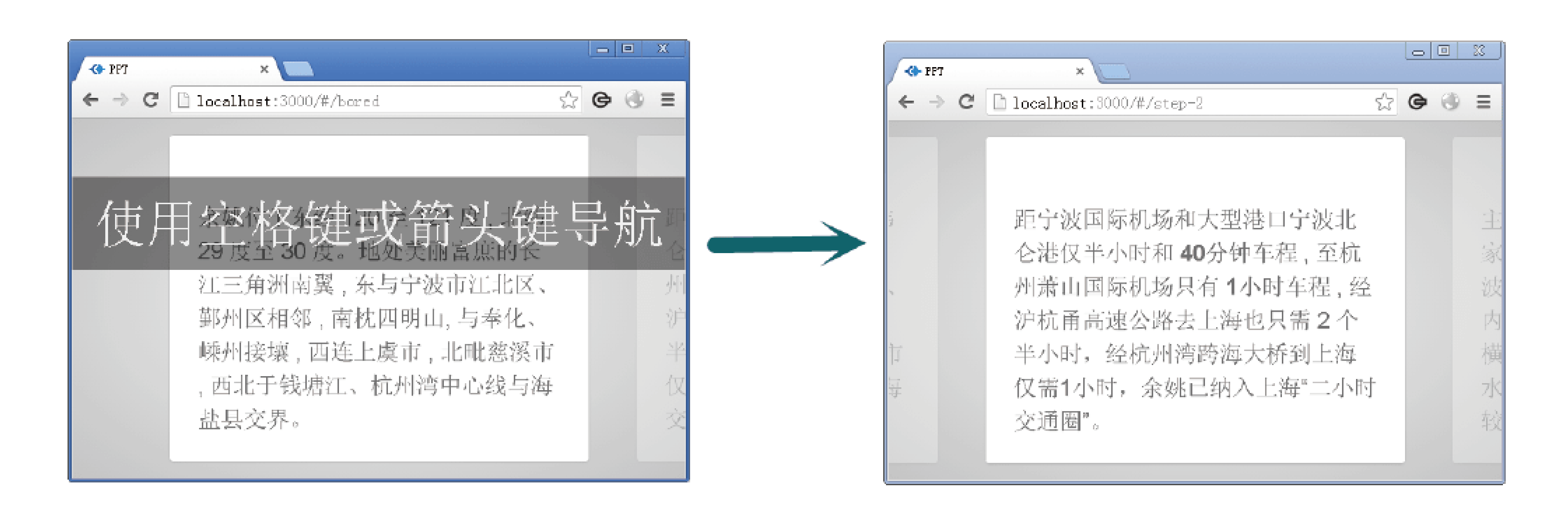


图13.5　PPT页面幻灯片向下一页切换



13.2 使用移动设备访问控制器页面

接下来，使用移动设备访问控制器页面，有两种方法可以访问部署页面：

- 第1种是将部署机器的IP和端口暴露给外网，手机可以直接访问对应的外网IP地址；
- 第2种是将手机和服务器连接在统一网关的网络，比如连接到同一台路由器。

笔者这里选择的是第2种方式，接着，获取服务器端的IP地址，可以通过在Windows下打开命令行输入获取，命令如下：

```
ipconfig
```

获取IP地址命令行提示信息如图13.6所示。

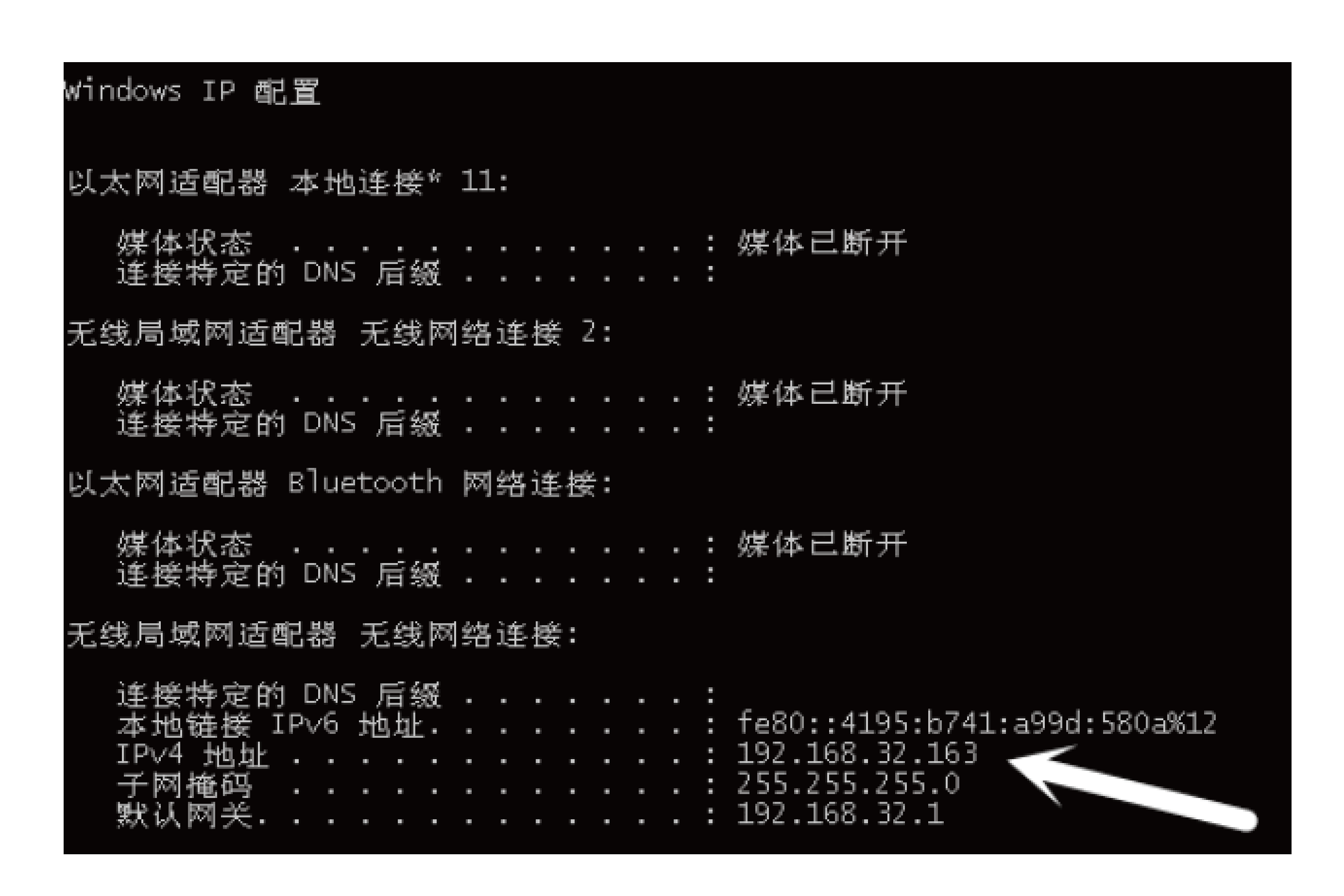


图13.6 当前机器的IP地址

下面通过手机浏览器访问控制器页面（笔者这里使用的手机为Android系统，采用默认自带的浏览器访问），通过ipconfig命令获取的服务器地址为192.168.32.163，则访问的控制器页面地址为http://192.168.32.163:3000/handle，效果如图13.7所示。

单指触碰屏幕并向右滑动，页面出现红色矩形框并向右移动（效果与鼠标操作PC端浏览器相似），如图13.8所示。

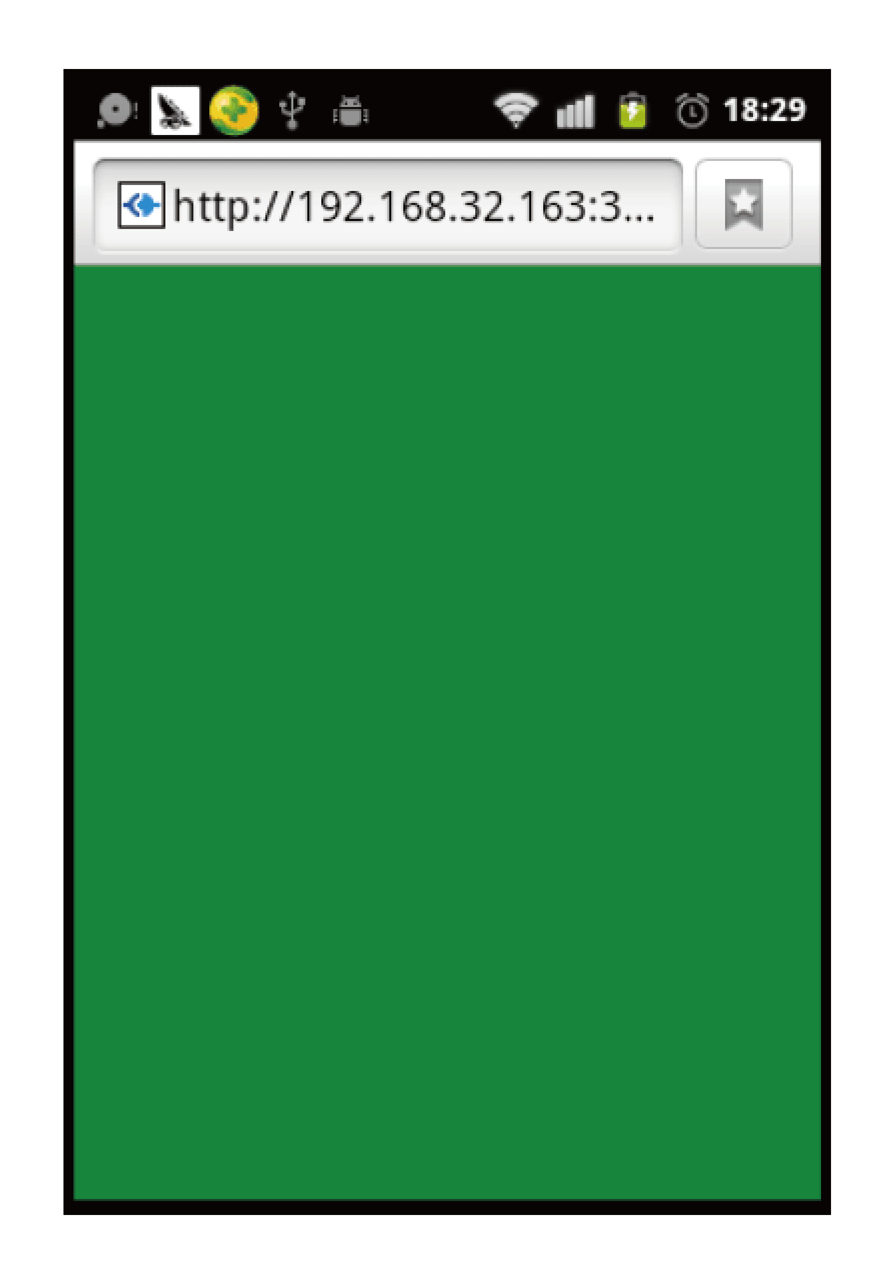



图13.7 手机浏览器访问控制器页面

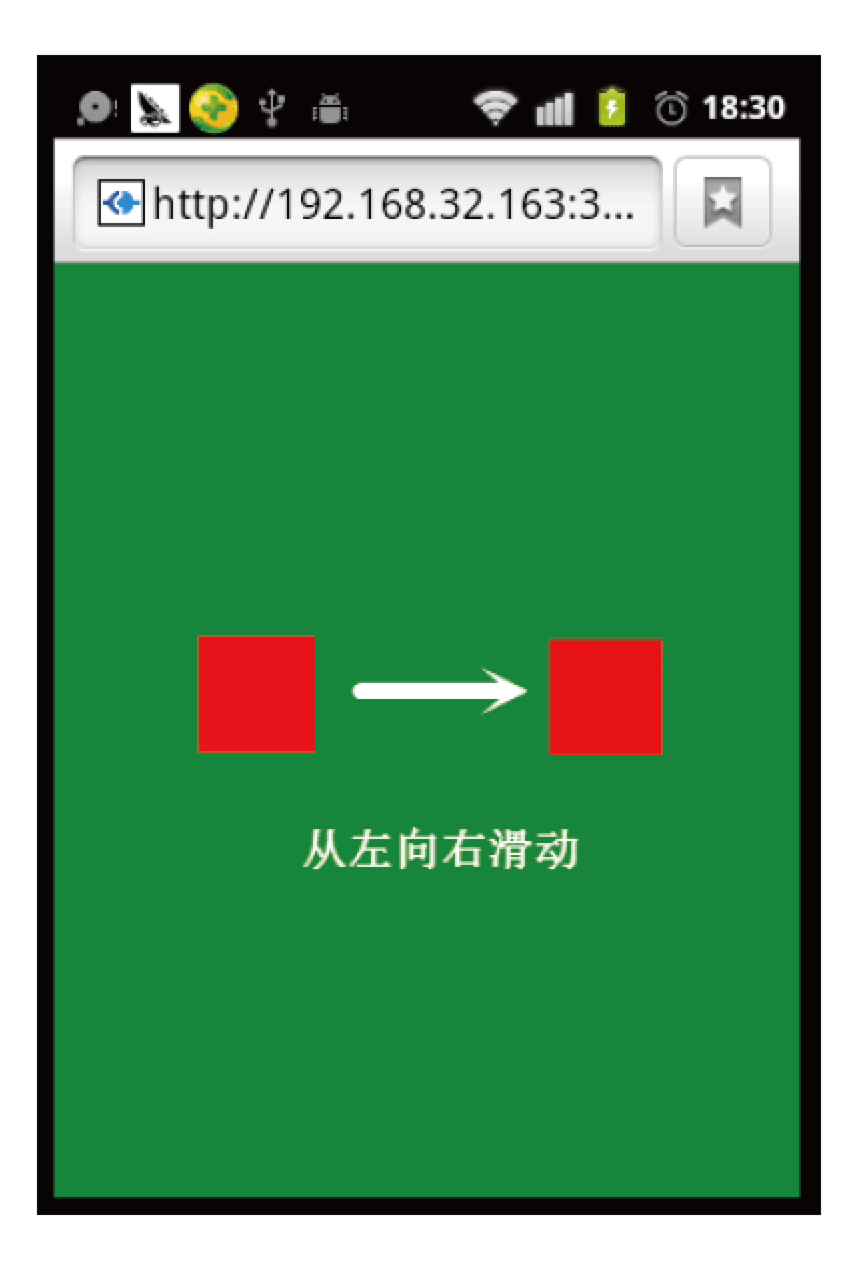


图13.8 触屏版操作控制器页面



本实例还有一个功能，用不同的终端访问PPT页面，可以通过同一个控制器来进行前后方翻页控制，同时保证各PPT页面同步操作运行，该功能通过HTML 5的WebSocket实现，利用了Socket.IO通信类库，下面的代码分析环节会介绍相关的技术要点。



13.3 代码设计和分析

13.3.1 启动服务器

首先看server.js脚本，该脚本主要负责启动Express框架搭建的Web服务器和Socket.IO类库搭建的Socket服务器，代码如下：

```
01 var express = require('express');        // 获取express模块用于构建Web服务
02 var routes = require('./routes');        // 获取routes模块用于页面路由
03 var http = require('http');      // 获取http模块处理请求信息和创建服务
04 var path = require('path');      // 获取path模块用于处理文件路径
05 var app = express();             // 新建一个express服务实例
06 var cons = require('consolidate'); // 获取consolidate模块，用于处理模板引擎
07 app.set('port', process.env.PORT || 3000);     // 设置服务器的端口号
08 app.set('views', __dirname + '/views');        // 设置服务器页面模板地址
09 app.engine('html', cons.swig);          // 设置swig模板作为服务器模板引擎
10 app.set('view engine', 'html');                // 设置模板引擎
11 app.use(express.favicon());                    // 设置页面图标
12 app.use(express.logger('dev'));                // 开启请求记录
13 app.use(express.bodyParser());   // 使用中间件bodyParser对请求包体进行解析
14 app.use(express.methodOverride());  // 处理POST请求伪装PUT等其他HTTP方法
15 app.use(app.router);                           // 设置启动路由功能
16 app.use(express.static(path.join(__dirname, 'public')));
   // 设置服务器静态文件目录
17 app.get('/', routes.index);                    // 配置'/'路由映射的逻辑模块
18 app.get('/handle',require('./routes/handle').handle);
   // 配置'/handle'路由映射的逻辑模块
19 var server = http.createServer(app);           // 创建一个Web服务器
20 server.listen(app.get('port'), function(){
   // 启动Web服务器并监听指定端口
21   console.log('Express server listening on port ' + app.get('port'));
22 });
23 var io = require('socket.io').listen(server);
   // 启动Socket服务并监听Web服务端口
24 io.sockets.on('connection', function (socket) { // 监听远端WebSocket接入
25   socket.emit('news', '欢迎加入WebSocket' );    // 向指定WebSocket发送消息
26   socket.on('handle', function (data) { // 监听自定义handle事件，并接收消息
27     io.sockets.emit('direction', data);
   // 向所有接入的socket连接发送自定义信息
28   });
29 });
```

代码第17行、第18行配置了PPT主页面和控制器页面的路由规则，即当访问的地址路径为对应的规则时执行对应的路由逻辑。

```
<style type="text/css">
  body { background-color: green; }                   <!-- 控制器底色 -->
  .move{                                         <!-- 拖动提示块样式 -->
    background-color: red;
    width: 50px; height: 50px;
    position: absolute;
    left: 0; top: 0;
    cursor: move;
  }
  p{ margin: 0;  padding: 0; }                   <!-- 操作日志列表样式 -->
</style>
</head>
<body>
  <div class="move" style="display:none"></div>       <!-- 拖动提示块 -->
</body>
<script src="socket.io/socket.io.js"></script>
<!-- 引入Socket.IO通信脚本 -->
<script>
  var socket = io.connect(location.origin);           // 建立socket连接
  var move_div = document.querySelector('div.move'), start;
  function log(msg){                          // log函数用于在页面上显示操作日志
    var p = document.createElement('p');
    p.innerHTML = msg;
    document.body.appendChild(p);
  };
  function get_postion(e , type){             // 根据事件类型获取鼠标当前的位置
    var pos={};
    if( e.type == type){                      // 判断mousedown、mousemove事件
      pos.x = e.pageX || e.clientX;                  // 获取鼠标的X轴位置
      pos.y = e.pageY;                               // 获取鼠标的Y轴位置
    }else{
      if(e.targetTouches.length >= 1){               // 元素内有手指触摸
        var touch = e.targetTouches[0];              // 获取手指列表的第一个
        pos.x = touch.pageX;                         // 获取手指触摸点的X轴位置
        pos.y = touch.pageY;                         // 获取手指触摸点的Y轴位置
      }
    };
    return pos;
  }
  function start(e){                                 // 拖动开始函数
    move_div.style.display = '';                     // 显示拖动提示方块
    start = get_postion(e,'mousedown').x;            // 记录拖动地点的X轴坐标
    document.addEventListener('mousemove',move,false);
    // 监听元素的鼠标移动事件
  };
  function end(e){                                   // 拖动结束函数
    move_div.style.display = 'none';
    var method , end = parseInt(move_div.style.left);
     // 获取方块的结束位置
    if(end > start){                          // 结束位置大于初始位置时，为向右滑动
      method = 'next';
    }else if (end < start){                   // 结束位置小于初始位置时，为向左滑动
      method = 'prev';
```

（1）当屏幕分辨率小于等于480px（手机）和767px（平板）时，原型如图14.1所示。主要包括可以查看阅读列表、收藏列表和喜欢列表的核心功能。



图14.1　原型图

（2）当屏幕分辨率大于767px（平板）小于1240px时。屏幕变大时，需要将第二重要的功能加入到页面中，根据设计原则，在767px~1240px范围内，加入“搜索”、“友情链接”、“查看会员”以及显示网站Logo等功能，原型设计图如图14.2所示。

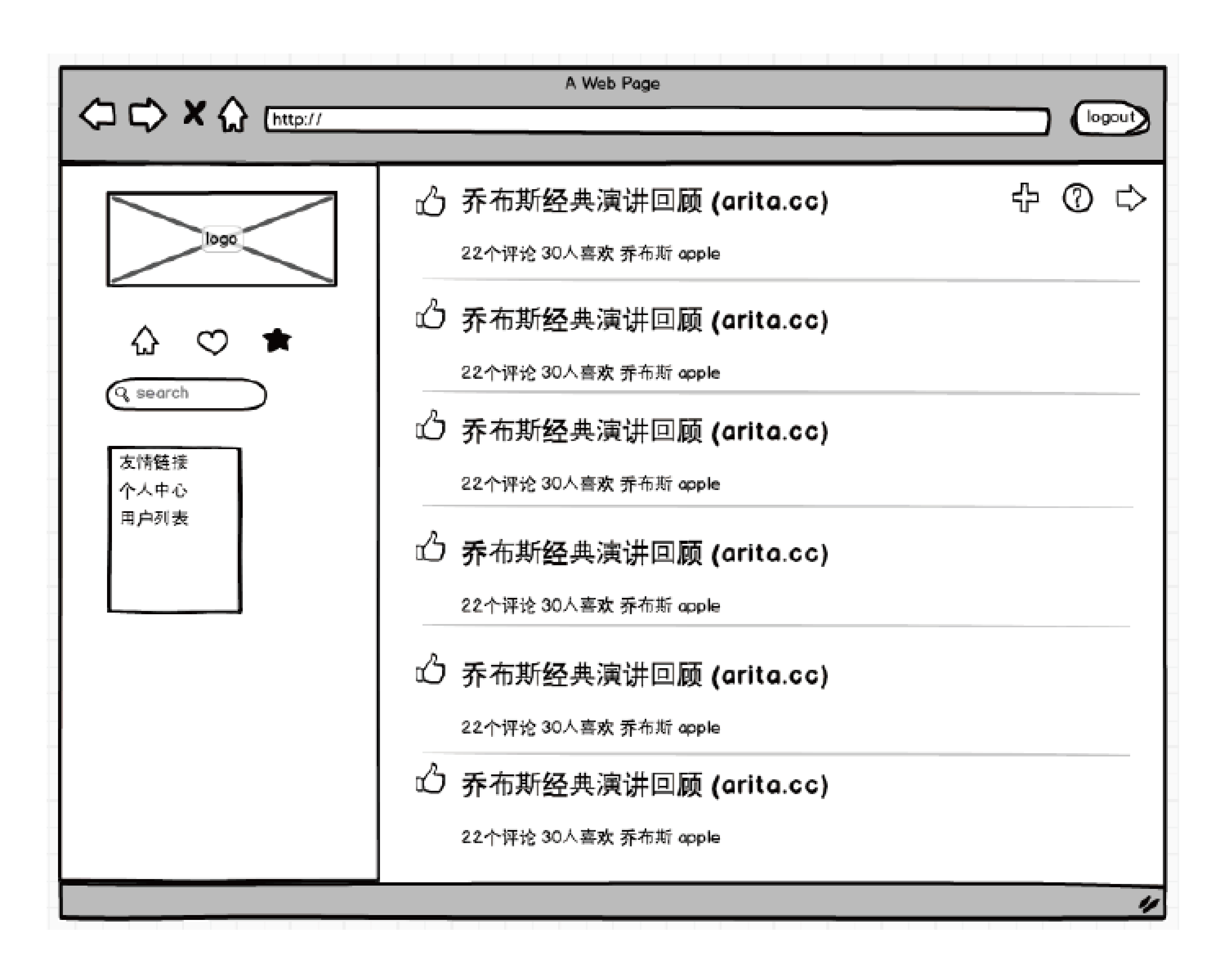


图14.2　宽度大于767px小于1240px时的原型图

（3）如果屏幕宽度大于1240px，则把右侧边栏也显示出来，原型设计如图14.3所示。

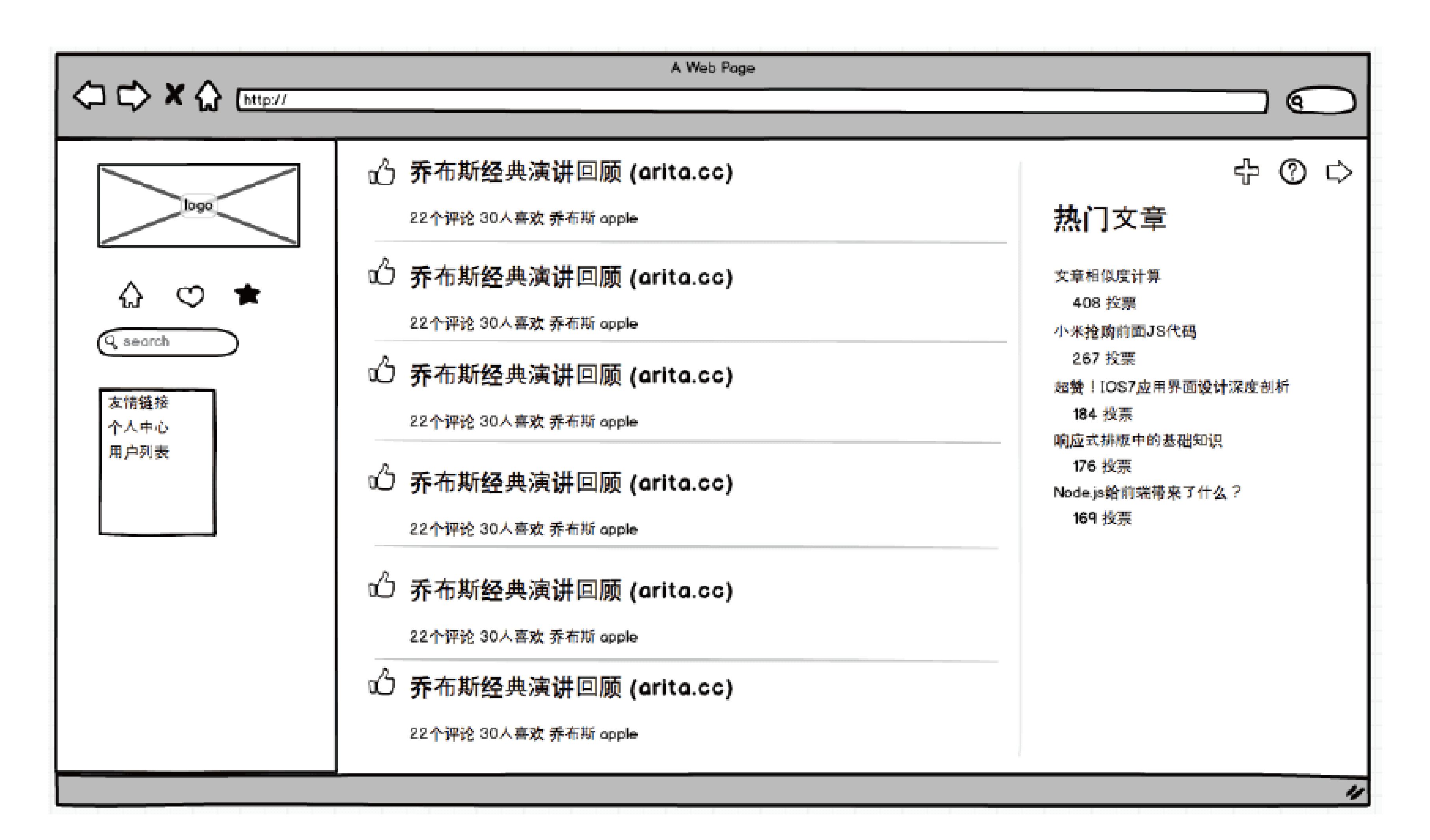


图14.3 屏幕宽度大于1240px时的原型图



14.2 模块设计

原型设计完成后，视觉设计师与前端开发工程师可以根据原型做模块设计。根据原型分析，主要模块包括图14.4所示的5个模块。

- 主体内容模块：该模块是最重要的内容，所以不管在任何屏幕分辨率下，均需要显示。
- 左侧导航模块：该模块提供左侧导航，当分辨率小于767px时，不显示。
- 移动版本页头模块：该模块是在分辨率小于767px时，提供的小屏导航区块，即小于767px时，左侧导航不显示，移动版本页头导航显示。
- 侧边栏：侧边栏属于辅助功能模块，仅当屏幕分辨率大于1240px，有足够的空间时才显示。
- 右上角功能区域：该区域提供辅助功能，虽然也是及其重要的功能，但在移动版本时，基本不会使用，所以当屏幕分辨率小于767px时，不显示。

模块设计分解效果如图14.4所示。

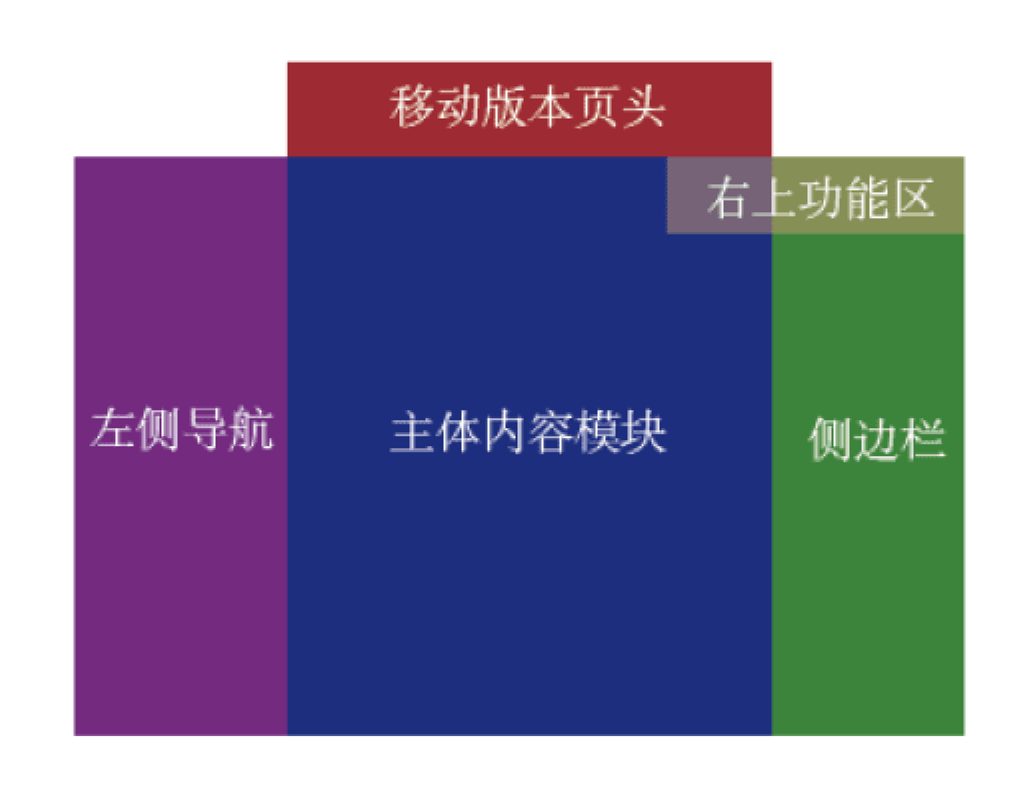


图14.4 模块设计分解图式

14.2.1 视觉模块设计

视觉设计师需要根据不同的分辨率给出不同的设计方案，但每个模块之间的过渡要求能够紧密衔接，不能产生视觉上的过于冲击的效果。因此需要有整体的视觉概念。视觉设计师需要根据480px、767px、979px、1240px、1500px的屏幕分辨率宽度设计出5个不同的版本。



在实际应用中，可能由于设计师资源等原因，来不及设计出5个不同的版本，而是3个版本，如767px、979px、1240px等。

14.2.2 前端模块设计

根据响应式的概念，每一个模块之间都要求能够互相解耦，依据这个原则，设计的5个前端模块代码如下：

```
<div id="top-menu"></div>                          <!-- 右上角顶部导航 -->
<div id="top-menu-mobile"></div>                   <!-- 移动版本顶部导航 -->
<div id="fixed-menu"></div>                        <!-- 左侧导航 -->
<div class="outer-wrap">
      <div id="main-container">                    <!-- 主体内容区域 -->
             <div id="main">
             </div>
      </div>
      <div id="sidebar">                           <!-- 右侧边栏 -->
             <div class="sidebar-inner" >
             </div>
      </div>
</div>
```

5个模块对应的CSS样式，具体解释见注释部分，代码如下：

```
#top-menu {                          <!-- 顶部右侧导航模块 -->
      position: fixed;               <!--采用fixed定位模式，保证其在右上角位置-->
      top: 5px;                      <!--距离最顶端5px -->
```

```
        right: 0;                  <!--使其靠在最右侧 -->
        z-index: 999;
        height: 29px;
        width: 140px;
        font-size: 20px;
        background-image: url("top-menu-bg.png");
}
#top-menu-mobile {                       <!-- 移动版本顶部导航模块 -->
        display: none;                   <!-- 该模块默认不显示 -->
        position: fixed; <!-- 采用fixed定位模式，保证其不随滚动而改变位置 -->
        top: 0px;                        <!-- 使其靠在最顶端 -->
        left: 0;                         <!-- 使其靠在最左边 -->
        z-index: 999;
        width: 100%;               <!-- 宽度设置为100%，使其充满整个宽度 -->
        height: 33px;
        padding-top: 10px;
        background-color: #282828;
        font-size: 24px;
        color: #fff;
        text-align: center;
}
#fixed-menu {                      <!--  左侧导航模块 -->
        background: #1d1d1d;
        position: fixed; <!-- 采用fixed定位模式，保证其不随滚动而改变位置 -->
        top: 0;                          <!-- 使其靠在最顶端 -->
        left: 0;                         <!-- 使其靠在最左边 -->
        width: 250px;
        z-index: 998;
        height: 100%;              <!-- 高度设置为100%，使其充满整个高度 -->
        overflow: hidden;
        border-right: 2px solid rgba(255,255,255,0.85);
        padding-top: 5px;
}
.outer-wrap {
        min-height: 100%;
        position: relative;
        margin-left: 260px;              <!-- 留出左侧260px给左侧导航 -->
        margin-top: 5px;
}
#main-container {                        <!-- 主体内容模块 -->
        float: left;
        width: 100%;
        margin-right: -330px;      <!-- 默认留出330px给右侧边栏模块 -->
        padding-bottom: 100px;
}
#main {
        background: #fff;
        margin-right: 330px;
        border-right: 1px solid rgba(0,0,0,0.1);
}
#sidebar {                               <!-- 右侧边栏区块 -->
```

```
        position: relative;
        float: left;
        padding: 23px 52px 0 10px;
        width: 328px;
        -webkit-box-sizing: border-box;
        -moz-box-sizing: border-box;
        box-sizing: border-box;
}
```

14.2.3 使用Media Queries自适应各种分辨率的客户端

媒体查询（Media Queries）是CSS 3的一个新特性，可以自动适应屏幕分辨率而使用不同的CSS样式，以达到自适应设备和屏幕分辨率的技术手段。以下是本实例页面的5个模块在不同分辨率下的不同响应，具体内容见代码注释部分，代码如下：

```
@media screen and (min-width: 1500px) {
<!-- 当屏幕分辨率大于1500px时，应用此样式 -->
        #main-container {
                max-width: 1240px;     <!-- 主体内容模块宽度最大为1240px -->
        }
        .inner-listing .main a:nth-child(2) {
                font-size: 18px !important;
<!-- 大分辨率，大字体，字体大小设置为18px -->
        }
}
@media screen and (max-width: 1240px) {
<!-- 当屏幕分辨率小于1240px时，应用此样式 -->
        #sidebar {
                display: none;              <!-- 右侧边栏模块不显示 -->
        }
        #main-container {
                margin-right: -120px;       <!-- 主题内容模块右间距调小 -->
        }
        #main {
                margin-right: 120px;
                border-right-style: none;
        }
}
@media (max-width: 979px) { <!-- 当屏幕分辨率小于979px时，应用此样式 -->
        .inner-listing .main a:nth-child(2) {
                font-size: 14px !important;  <!-- 字体大小从18px变为14px -->
                max-width: 95% !important;  <!-- 主体内容模块列表页宽度变小 -->
        }
        .outer-wrap {
                margin-top: 20px;
        }
        #main-container {
                margin-right: 0;     <!-- 主题内容模块右间距设为0 -->
        }
        #main {
```

```
                margin-right: 0;
        }
}
@media (max-width: 767px) {    <!-- 当屏幕分辨率小于767px时，应用此样式 -->
        body {
                padding-right: 5px;
                padding-left: 10px;
        }
        #fixed-menu {
                display: none;              <!-- 左侧导航模块隐藏，不显示 -->
        }
        #top-menu {
                display: none;              <!-- 顶部右侧导航模块不显示 -->
        }
        #top-menu-mobile {
                display: block;             <!-- 显示移动版本顶部导航模块 -->
        }
        #top-stripe { position: fixed; top: 43px; left: 0; }
        .outer-wrap { margin-left: 0; margin-top: 48px; }
        #main-container { margin-right: 0; }
        #main { margin-right: 0; }
        #back-to-top { display: none !important; }
}
@media (max-width: 480px) {    <!-- 当屏幕分辨率小于480px时，应用此样式 -->
        .article .inner-listing {
                margin-left: 0 !important;
<!-- 主体内容模块列表页间距设置为0 -->
                margin-right: 0 !important;
        }
}
```



本实例具体的代码见光盘文件。



14.3 运行效果

本实例在Chrome浏览器下运行通过，请使用Chrome浏览器打开本实例，在1500以上的分辨率下运行，效果如图14.5所示。



图14.5　1500以上分辨率运行效果



本实例还采用了“@font-face”技术来实现各种图标。

手动拉动浏览器宽度，当宽度缩小到1239px时，右侧边栏隐藏，如图14.6所示。



图14.6　右侧边栏隐藏

继续拉动浏览器，当浏览器宽度达到766px时，隐藏左侧导航和右上角导航，显示移动版本顶部导航，如图14.7所示。

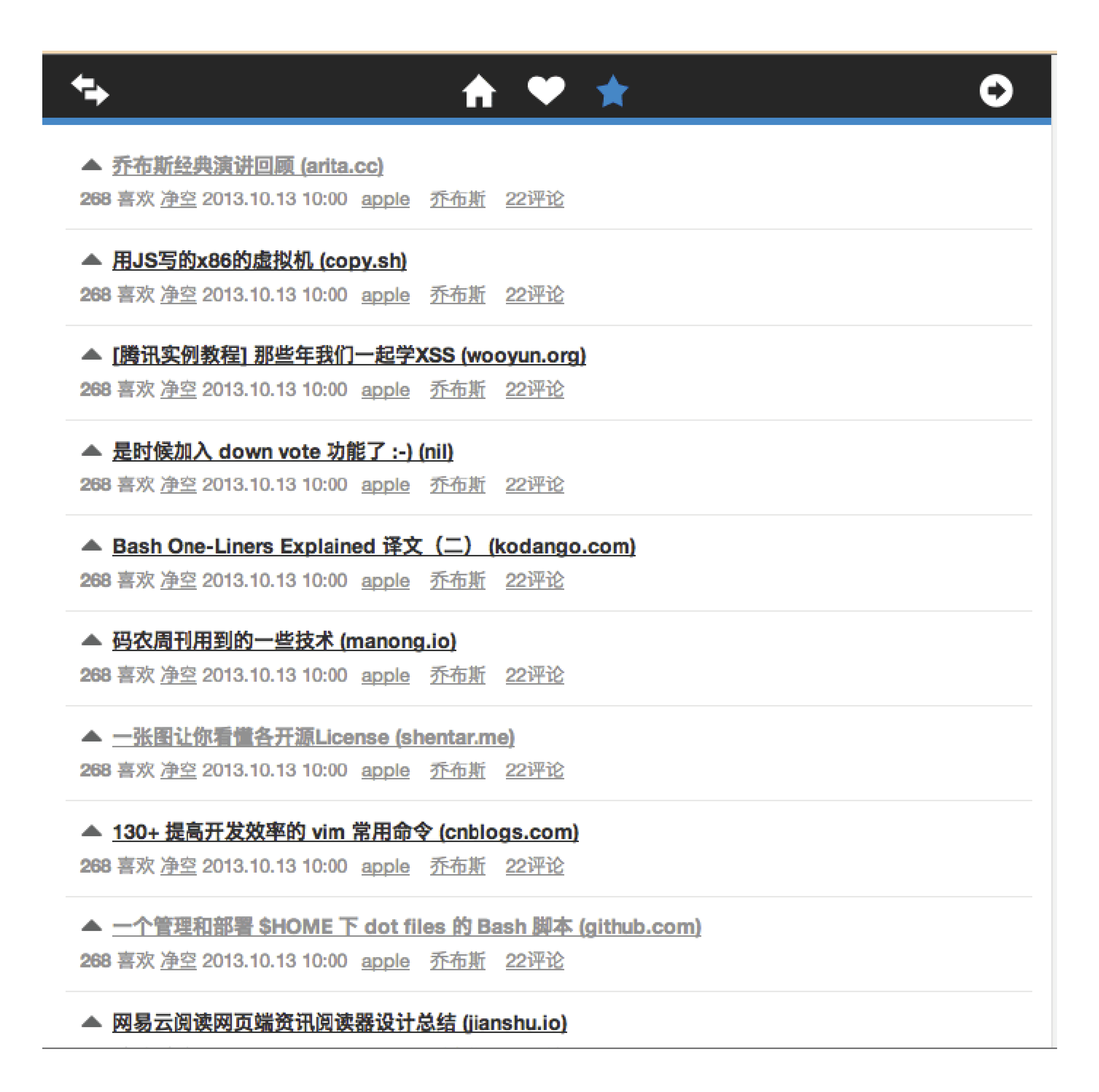


图14.7 平板电脑下自适应响应效果

继续缩小浏览器宽度，到达480px时，效果如图14.8所示。

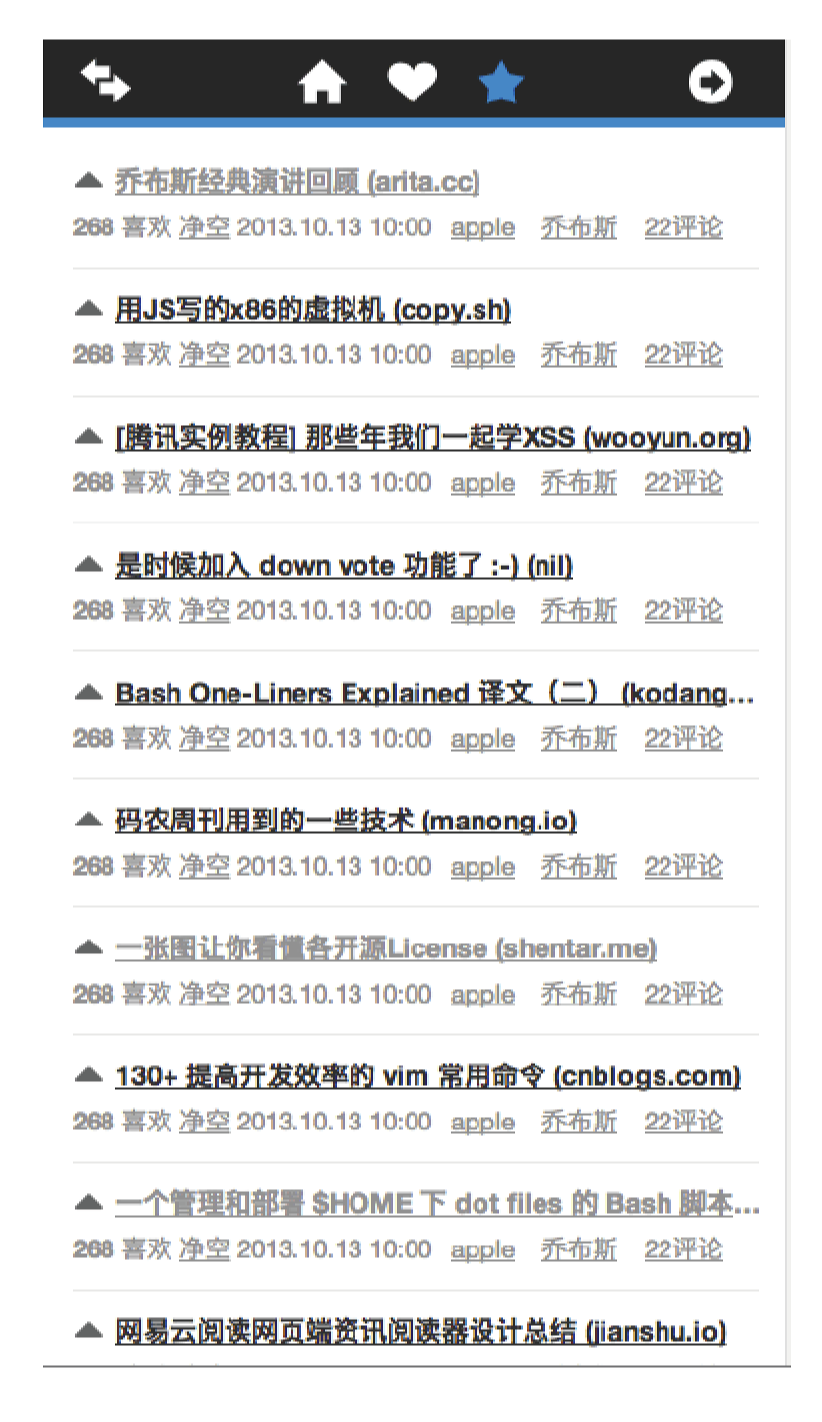


图14.8 手机浏览器下的显示效果



14.4 本章小结

响应式设计已经成为一种设计趋势，而响应式设计的一般设计流程是需求分析、原型设计、模块设计（视觉设计、前端模块设计），设计原则一般是遵行“设计先行、内容优先、移动优先”的原则。响应式的Web设计优势明显，只需要开发一种页面，并针对于不同的分辨率、不同的设备环境进行设计，在开发、维护和运营成本上，相对于传统的多种版本的开发，是绝对的优势。但同时，对于传统的互联网交互设计和前端实现来说，提出了更加高的要求，设计之初就需要考虑清楚不同分辨率下的页面布局变化。本章响应式案例的介绍，只是一个开始，读者还需要通过大量的练习才能掌握响应式设计的精髓。

附录A 主流浏览器对HTML 5新特性的支持情况

附表A.1 CSS 3属性（√表示支持，×表示不支持）

属性	Chrome 25	Firefox 15	Opera 12	IE 6	IE 7	IE 8	IE 9	IE 10
RGBA	√	√	√	×	×	×	√	√
HSLA	√	√	√	×	×	×	√	√
Box Sizing	√	√	√	×	×	√	√	√
Background Size	√	√	√	×	×	×	√	√
Multiple Backgrounds	√	√	√	×	×	×	√	√
Border Image	√	√	√	×	×	×	×	×
Border Radius	√	√	√	×	×	×	√	√
Box Shadow	√	√	√	×	×	×	√	√
Text Shadow	√	√	√	×	×	×	×	√
Opacity	√	√	√	×	×	×	√	√
CSS Animations	√	√	√	×	×	×	×	√
CSS Columns	√	√	√	×	×	×	×	√
CSS Gradients	√	√	√	×	×	×	×	√
CSS Reflections	√	×	×	×	×	×	×	×
CSS Transforms	√	√	√	×	×	×	√	√
CSS Transforms 3D	×	√	×	×	×	×	×	√
CSS Transitions	√	√	√	×	×	×	×	√
CSS FontFace	√	√	√	√	√	√	√	√
FlexBox	√	√	×	×	×	×	×	×
Generated Content	√	√	√	×	×	√	√	√
DataURI	√	√	√	×	×	√	√	√
Pointer Events	√	√	×	×	×	×	×	×
Display: table	√	√	√	×	×	√	√	√
Overflow Scrolling	×	×	×	×	×	×	×	×
Media Queries	√	√	√	×	×	×	√	√

附表A.2 CSS 3选择器（√表示支持，×表示不支持）

属性	Chrome 25	Firefox 15	Opera 12	IE 6	IE 7	IE 8	IE 9	IE 10
Begins with	√	√	√	×	√	√	√	√
Ends with	√	√	√	×	√	√	√	√
Matches	√	√	√	×	√	√	√	√
Root	√	√	√	×	×	×	√	√
nth-child	√	√	√	×	×	×	√	√
nth-last-child	√	√	√	×	×	×	√	√

（续表）

属性	Chrome 25	Firefox 15	Opera 12	IE 6	IE 7	IE 8	IE 9	IE 10
nth-of-type	√	√	√	×	×	×	√	√
nth-last-of-type	√	√	√	×	×	×	√	√
last-child	√	√	√	×	×	×	√	√
first-of-type	√	√	√	×	×	×	√	√
last-of-type	√	√	√	×	×	×	√	√
only-child	√	√	√	×	×	×	√	√
only-of-type	√	√	√	×	×	×	√	√
empty	√	√	√	×	×	×	√	√
target	√	√	√	×	×	×	√	√
enabled	√	√	√	×	×	×	√	√
disabled	√	√	√	×	×	×	√	√
checked	√	√	√	×	×	×	√	√
not	√	√	√	×	×	×	√	√
General Sibling	√	√	√	×	√	√	√	√

附表 A.3 HTML 5 Web应用程序（√表示支持，×表示不支持）

属性	Chrome 25	Firefox 15	Opera 12	IE 6	IE 7	IE 8	IE 9	IE 10
Local Storage	√	√	√	×	×	√	√	√
Session Storage	√	√	√	×	×	√	√	√
Post Message	√	√	√	×	×	√	√	√
Offline Applications	√	√	√	×	×	×	×	√
Workers	√	√	√	×	×	×	×	√
Query Selector	√	√	√	×	×	√	√	√
WebSQL Database	√	×	√	×	×	×	×	×
Indexed Database	√	√	×	×	×	×	×	√
Drag and Drop	√	√	√	√	√	√	√	√
Hash Change (Event)	√	√	√	×	×	√	√	√
History Management	√	√	√	×	×	×	×	√
WebSockets	√	√	×	×	×	×	×	√
GeoLocation	√	√	√	×	×	×	√	√
Touch	×	×	×	×	×	×	×	×
File API	√	√	√	×	×	×	×	√
Meter element	√	×	√	×	×	×	×	×
Progress element	√	√	√	×	×	×	×	√

附表 A.4 HTML 5图形与嵌入式内容（√表示支持，×表示不支持）

属性	Chrome 25	Firefox 15	Opera 12	IE 6	IE 7	IE 8	IE 9	IE 10
Canvas	√	√	√	×	×	×	√	√
Canvas Text	√	√	√	×	×	×	√	√
SVG	√	√	√	×	×	×	√	√
SVG Clipping Paths	√	√	√	×	×	×	√	√
SVG Inline	√	√	√	×	×	×	√	√
SMIL	√	√	√	×	×	×	× √	√
WebGL	√	√	√	×	×	×	√	√
Audio	√	√	√	×	×	×	√	√
Video	√	√	√	×	×	×	√	√

附表 A.5 HTML 5音频编解码器（√表示支持，×表示不支持）

属性	Chrome 25	Firefox 15	Opera 12	IE 6	IE 7	IE 8	IE 9	IE 10
Audio: ogg/vorbis	√	√	√	×	×	×	×	×
Audio: mp3	√	×	×	×	×	×	√	√
Audio: wav	√	√	√	×	×	×	×	×
Audio: AAC	√	×	×	×	×	×	√	√

附表 A.6 HTML 5视频编解码器（√表示支持，×表示不支持）

属性	Chrome 25	Firefox 15	Opera 12	IE 6	IE 7	IE 8	IE 9	IE 10
Video: ogg/theora	√	√	√	×	×	×	×	×
Video: H.264	√	×	×	×	×	×	√	√
Video: WebM	√	√	√	×	×	×	×	×

附表 A.7 HTML 5表单输入（√表示支持，×表示不支持）

属性	Chrome 25	Firefox 15	Opera 12	IE 6	IE 7	IE 8	IE 9	IE 10
Form: Search	√	√	√	×	×	×	×	√
Form: Phone	√	√	√	×	×	×	×	√
Form: URL	√	√	√	×	×	×	×	√
Form: Email	√	√	√	×	×	×	×	√
Form: DateTime	×	×	√	×	×	×	×	×
Form: Date	√	×	√	×	×	×	×	×
Form: Month	√	×	√	×	×	×	×	×
Form: Week	√	×	√	×	×	×	×	×
Form: Time	√	×	√	×	×	×	×	×
Form: LocalTime	√	×	√	×	×	×	×	×
Form: Number	√	×	√	×	×	×	×	√
Form: Range	√	×	√	×	×	×	×	√
Form: Colour	√	×	√	×	×	×	×	×

附表A.8 HTML 5表单属性（√表示支持，×表示不支持）

属性	Chrome 25	Firefox 15	Opera 12	IE 6	IE 7	IE 8	IE 9	IE 10
Form: Autocomplete	√	√	√	×	×	×	×	√
Form: Autofocus	√	√	√	×	×	×	×	√
Form: List	√	√	√	×	×	×	×	√
Form: Placeholder	√	√	√	×	×	×	×	√
Form: Min	√	×	√	×	×	×	×	√
Form: Max	√	×	√	×	×	×	×	√
Form: Multiple	√	√	√	×	×	×	×	√
Form: Pattern	√	√	√	×	×	×	×	√
Form: Required	√	√	√	×	×	×	×	√
Form: Step	√	×	√	×	×	×	×	√



数据来源于网站http://fmbip.com/litmus。

附录B 传统HTML标签及说明

为了让读者可以轻松过渡和对比，这里列出了传统HTML 4标签的说明。

附表B.1 文件基本标签和页面设置标签

标签	说明
HTML	文件类型标签，表示这是一个HTML文件
HEAD	文件头标签，标记HTML文件头部信息的开始和结束
NAME="keywords"	在头部信息中设置页面的关键字
NAME="discription"	在头部信息中设置页面的描述文字
NAME="generator"	在头部信息中设置编辑工具
NAME="author"	在头部信息中设置作者信息
NAME="robots"	在头部信息中设置限制搜索的方式
NAME="copyright"	在头部信息中添加版权声明
NAME="build"	在头部信息中设置页面的创建日期
NAME="reply-to"	在头部信息中添加联系人的邮箱
HTTP-EQUIV="Content-Language"	在头部信息中标明页面使用的语言
HTTP-EQUIV="refresh"	在头部信息中设置网页的定时刷新和跳转
HTTP-EQUIV="expires"	在头部信息中设置网页的到期时间
HTTP-EQUIV="cache-control"	在头部信息中设置禁止从缓存中调用该页面
HTTP-EQUIV="set-cookie"	删除过期的cookie
HTTP-EQUIV="window-target"	在头部信息中设置该页面以独立窗口打开
HTTP-EQUIV="过渡事件”	在头部信息中设置网页的过渡效果
BASE	基底网址标签，用来设置浏览器中文件根目录的绝对路径
LINK	页面相关链接，即需要使用外部文件时，指明链接文件与页面的位置关系、链接文件的名称和地址等内容
TITLE	页面标题标签，用于显示页面的标题，其内容显示在浏览器的标题栏
BODY	文件主体标签，表示文件的主体内容，如页面中真实显示的内容
BGCOLOR	设置页面的背景颜色，一般需要在文件的主体标签内说明，即在BODY标签中使用，如<BODY BGCOLOR="red">表示页面背景是红色
BACKGROUND	设置页面的背景图像，当该标签和BACKCOLOR一起使用时，其优先级将高于BACKCOLOR标签
BGPROPERTIES	用于设置背景图像是否随着滚动条的运动而变化
TEXT	设置页面的默认文字颜色
LEFTMARGIN	设置页面的左边距
TOPMARGIN	设置页面上边距

附表B.2 超级链接的相关标签

标签	说明
A	设置页面中的超级链接，表示链接内容的开始和结束
HREF	设置超级链接的链接地址，可以是绝对地址和相对地址
TARGET	用于设置目标页面的打开方式

（续表）

标签	说明
NAME	在网页中添加一个锚点，便于在链接中找到相应的内容
#	用于HREF属性的值中，表示将要链接到的内容是页面中的某个锚点
LINK	设置链接文字的初始颜色
ALINK	设置使用中，即正在访问的链接的文字颜色
VLINK	修改访问过的链接文字颜色

附表B.3 页面文字的相关标签

标签	说明
H	设置为不同级别的标题文字，标题的数值越小，其文字反而越大，如H1是一级标题，其文字最大
ALIGN	跟在H标签的后面使用，用于设置标题文字的对齐方式
I、EM	将文字设置为斜体样式
B、STRONG	用于设置文字的加粗
U	为文字添加下划线
S	在文字上添加删除线的效果
SUB、SUP	将文字设置为上标或下标的样式
TT	设置文字为打字机字体
CITE	将文字设置为引文格式
CODE	将文字设置为程序码样式
KBD	将文字设置为键盘输入样式
VAR	将文字设置为变量声明的样式
BIG、SMALL	字体加大和缩小
BLOCKQUOTE	设置文本的缩进，可以嵌套使用
SAMP	样本字体，使文字等宽显示
DFN	表示定义或说明性文字，一般以斜体字表示
ADDRESS	地址标签，一般用于标签表示地址的内容，如电子邮件地址
BASEFONT SIZE	用于设置基本字的大小
FONT	用于设置文字的属性，一般需要和其他一些属性设置标签同时使用，如SIZE、COLOR等
PRE	预定义格式标签，表示HTML页面中将完全按照输入的效果来显示这段文字
CENTER	设置其中的内容为居中对齐
BLINK	设置文字的闪烁效果
	特殊字符空格，由于HTML自动忽略多个空格，因此需要输入多个空格时，可以多次使用该标签
&和;组合	在HTML中利用“&”标签做前缀，利用“;”标签作后缀，可以显示一些特殊字符，如“&grade”表示注册商标的标签符号

附表B.4 图像与多媒体

标签	说明
IMG	图像标签，需要和图像的其他属性同时使用，如SRC等
SRC	设置图像的源文件地址，在IMG标签中使用
WIDTH、HEIGHT	设置图像显示的宽度和高度，防止图像过大或者过小
ALIGN	设置图像与同行文字的对齐关系，包括top、middle、left等值
ALT	设置当图像不能正常显示时，页面中显示出来的替换文字
BORDER	为图像设置边框

（续表）

标签	说明
HSPACE、VSPACE	设置图像与文字相连时的水平间隔和垂直间隔，可以防止页面布局过于拥挤
LOWSRC	设置低解析度图像，便于用户能很快地看到图像的效果
MAP	应用图像地图，即通过图像设置页面中的超级链接，同时需要设置图像的热区形状和热区坐标，一般在可视化设计的软件中能够更方便地使用，如Dreamweaver
BGSOUND	为页面添加背景音乐，需要和SRC属性一起使用
SRC	和BGSOUND属性同时使用，表示背景音乐的文件地址
LOOP	与BGSOUND属性同时使用，表示背景音乐的循环播放次数
EMBED	为页面添加多媒体元素，可以是Flash动画、AVI文件等
AUTOSTART	在EMBED标签内使用，表示多媒体文件是否自动播放
LOOP	在EMBED标签内使用，表示多媒体文件是否会循环播放
HIDDEN	在EMBED标签内使用，用于隐藏多媒体文件的播放面板
MARQUEE	添加滚动文字
DIRECTION	与MARQUEE标签一起使用，可以设置文字的滚动方向
BEHAVIOR	与MARQUEE标签一起使用，设置文字的滚动方式
SCROLLAMOUNT	与MARQUEE标签一起使用，调整文字滚动速度
SCROLLDELAY	与MARQUEE标签一起使用，设置滚动延迟
LOOP	与MARQUEE标签一起使用，设置文字循环滚动的次数
BGCOLOR	与MARQUEE标签一起使用，为滚动文字设置背景色
WIDTH、HEIGHT	与MARQUEE标签一起使用，设置滚动文字的背景面积
HSPACE、VSPACE	与MARQUEE标签一起使用，设置滚动文字的空白空间

附表B.5 页面布局

标签	说明
DIV	层标签，使层内的元素作为一个整体出现
ALIGN	跟在DIV标签后面使用，表示层的对齐方式
STYLE	层的样式设置
P	段落标签
ALIGN	设置段落中的文字对齐方式
BR	换行标签，连续使用可以多次换行
NOBR	设置文字在页面中不进行自动换行
WBR	强制换行标签，在使用了NOBR标签后，可以通过该标签对文字进行强制性的换行操作
HR	插入水平线
SIZE	位于HR标签内使用，用于设置水平线高度
WIDTH	位于HR标签内使用，用于设置水平线宽度
ALIGN	位于HR标签内使用，用于设置水平线的对齐方式
NOSHADE	位于HR标签内使用，将添加的水平线设置为实线效果
COLOR	位于HR标签内使用，用于设置水平线的颜色

附表B.6 列表

标签	说明
UL	无序列表标签
OL	有序列表标签
TYPE	在列表的标签内使用，用于设置列表的序号类型。在不同类型的列表中，其具体含义也不相同
START	在OL标签内使用，用于设置有序列表的起始数值

（续表）

标签	说明
DL	定义列表标签
MENU	菜单列表标签
DIR	目录列表

附表B.7 表单

标签	说明
FORM	添加表单
ACTION	设置表单的处理程序
NAME	在FORM标签内使用，可以设置这个表单的名称
METHOD	设置表单信息的传送方法
ENCTYPE	设置表单的编码方式，如纯文本方式发送
INPUTTYPE	设置表单中的控件类型，通过不同的取值来添加不同的控件，如设置该属性值为text，则添加一个文本框控件
INPUTNAME	为表单中的控件设置名称
INPUTVALUE	为表单中的控件设置默认值
CHECKED	将表单中的某个控件设置为已选定的选项
SIZE	设置控件的长度，用在文本框、密码域以及文件域控件中
MAXLENGTH	设置控件中可以输入的最大的字符数，例如设置为6，则输入了6个字符以后就无法再继续输入了
SELECT	添加一个下拉式的选单控件，如菜单、列表
SIZE	设置选单显示的项目数，如列表中包含的列表项数目
OPTION	设置选单控件中的选单项目，每一个OPTION标签就表示一个选单项，因此设置SIZE属性为多少，就应该有多少个OPTION标签
NAME	在SELETE标签内使用，可以设置选单控件的名称
VALUE	用于设置选单控件中各个选项的值
SELECTED	设置该选单项默认情况下已被选中
MULTIPLE	用于设置列表项目，在SELETE标签中添加该标签，表明这个选单控件是一个多选控件，用户可以同时选中多个选项
TEXTAREA	添加一个文本输入区域控件，一般用来创建能够输入多行文本的区域
NAME	在TEXTAREA标签内使用，可以设置该文本区域的名称
ROWS、COLS	在TEXTAREA标签内使用，用于设置输入区的大小，即该输入区域显示在页面的内容包含多少行和多少列
WRAP	用于设置输入区的换行方式，即是否自动换行

附表B.8 表格

标签	说明
TABLE、TR、TD	表示在页面创建一个表格，在其中将包括TR和TD标签，TR标签用于设置表格中的行数，而TD则表示表格中的列
BORDER	设置表格的边框粗细，如果设置为0，则不显示边框
BORDERCOLOR	设置表格的边框颜色
BORDERCOLORLIGHT	设置表格的亮边框颜色
BORDERCOLORDARK	设置表格的暗边框颜色
WIDTH	在TABLE标签内使用，可以设置表格的显示宽度。另外，该标签也可以用在TR和TD标签内，用于设置行或者单元格的宽度

（续表）

标签	说明
HEIGHT	在TABLE标签内使用，可以设置表格的高度；该标签同样可以用于TR和TD标签内
BGCOLOR	在TABLE标签内使用，设置表格背景色
BACKGROUND	在TABLE标签内使用，可以设置表格的背景图像，它的优先级高于BGCOLOR标签
CELLSPACING	在TABLE标签内使用，可以设置表格内框的宽度
CELLPADDING	在TABLE标签内使用，可以设置文字与边框之间的距离
ALIGN	在TABLE标签内使用，可以设置表格的对齐方式
COLSPAN	设置单元格的水平跨度
ROWSPAN	设置单元格的垂直跨度
NOWRAP	用在TD标签内，设置单元格内的文字内容不自动换行
THEAD	为表格设置表首，常常与TH标签一起使用，用于突出显示表格的首行，如列名、数据项名称；也可以与COLSPAN标签合用，对表格进行内容分类
TBODY	为表格设置主体区域
TFOOT	为表格设置表尾
TH	为表格设置表头
CAPTION	为表格添加标题单元格，一般设置在表格上方
ALIGN	设置表格标题文字的水平对齐方式
VALIGN	设置表格标题文字的垂直对齐方式

附表B.9 框架标签

标签	说明
FRAMESET	在页面中设置框架，框架中需要包含对象，即框架文件FRAME。一般来说，FRAMESET标签中包含了几个对象，就要有几个FRAME标签
ROWS	将框架窗口上下排列，并设置各个窗口显示的比例
COLS	将框架窗口左右排列，并设置各个窗口显示的比例
FRAMEBORDER	在FRAMESET标签内使用，设置框架窗口的边框
FRAMESPACING	在FRAMESET标签内使用，设置框架的边框宽度
BORDERCOLOR	在FRAMESET标签内使用，设置框架的边框颜色
SRC	在FRAME标签内使用，设置框架窗口的页面源文件
NAME	在FRAME标签内使用，设置框架窗口的名称，一般用于区分各个框架窗口
SCROLLING	在FRAME标签内使用，设置框架滚动条的显示效果
MARGINWIDTH	在FRAME标签内使用，设置框架的边缘宽度
MARGINHEIGHT	在FRAME标签内使用，设置框架的边缘高度
NORESIZE	将框架设置为禁止调整窗口的尺寸
NOFRAMES	设置页面不支持框架标签，这样当在不支持框架页面的浏览器中显示页面时，可以自动忽略其中的内容
IFRAME	插入浮动框架
SRC	在IFRAME标签内使用，设置浮动框架的页面源文件
WIDTH、HEIGHT	在IFRAME标签内使用，设置浮动框架的大小
ALIGN	设置浮动框架的对齐方式
FRAMEBORDER	设置浮动的边框显示属性
MARGINWIDTH	设置浮动框架边缘的宽度
MARGINHEIGHT	设置浮动框架边缘的高度
SCROLLING	设置浮动框架滚动条的效果